2011

Gobuddy - Android mobile application

Kalaivani Nellaiappan

GOBUDDY - ANDROID MOBILE APPLICATION

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Kalaivani Nellaiappan

March 2011

GOBUDDY - ANDROID MOBILE APPLICATION

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Kalaivani Nellaiappan

March 2011

Approved by:

[redacted]

Dr. Ernesto Gomez, Advisor, Computer Science & Engineering

3/8/2011

Date

[redacted]

Dr. Kerstin Voigt

[redacted]

Dr. Tong Yu

ABSTRACT

GoBuddy is a mobile application that has been developed for Android based smart phones. The application features location based services to the end user. The core functionality is to deliver a rich set of information about a particular place of interest. It operates in two different modules. One is to provide the end user with the information about common places of interest such as coffee shops, restaurants, fast foods, gas stations, hotels, movie theaters, and attractions in and around the user's current location, where the handheld is used. The application senses the user's current latitude and longitude using the in-built GPS on the smart phone and provides detailed information about a particular place such as address and rating, map view showing the exact GeoPoint of the particular place, directions and distance to the place from the current GeoPoint. The second module allows the user to search his own place of interest in the current location and also in any other location within the United States. In addition to being an information repository, it also has features such as direct one touch calling, navigating to the URL of the place if any, and providing multiple views of the map such as satellite views, street views, and traffic conditions. The developed

application ensures richness of data and ease of use making it very comfortable for the end user.

ACKNOWLEDGMENTS

I would like to thank God for being with me and giving me the strength to complete my project and the Master's degree. My heartfelt thanks to my parents Nellaiappan and Parvathi, who have been with me for most of the time in the past two years, for their blessings and for taking care of my kids. I would like to thank my husband Sundar for his encouragement and his moral support throughout. I would like to thank my children Keerthana and Krithik for being very patient with me during the entire course of study. I would like to thank my friends, who have cheered me and constantly kept my spirits up. I would like to extend my sincere thanks to the faculty of the Department of Computer Science and Engineering, CSUSB Dr. Ernesto Gomez, Dr. Kerstin Voigt, and Dr. Tong Yu for their valuable advice, help and support in completing the project. It has been a great experience learning at the University. The learning experience I gained through the scholarly professors was tremendous. Finally, I would like to thank Dr. Josephine Mendoza, Graduate Coordinator, for her valuable guidance, without which it would have been impossible for me to earn the Master's degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

With the immensely growing technology, people have grown from getting information from paper-based elements to one-touch electronic ensembles. Everyone wants to gain access to the information at anytime and anywhere. This has led to many innovative inventions, with mobile applications dominating the trend today. Mobile applications are accessible at all times, irrespective of the place and time. Mobile applications really go handy and serve better starting from simple applications to complex ones. One of the most common applications, that every mobile user would need at some point is an application that would give all the information about a place he is at the moment or planning to visit a place in course of time. The basic idea of this project is to develop an Android application that provides location based services specific to a place. This application has been developed for the general audience and integrates various essential features that correspond to a particular location in the United States.

1.1 Purpose of Project

The purpose of this application is to serve the end user, with reliable, instantaneous and location based information in an easy and much user friendly way. The basic information includes viewing the map and address of the place of interest and getting the directions to a particular place in addition to having some extra features. The application provides both map based and text based information about pre-listed places that are in close proximity to the user such as coffee shops, restaurants, fast foods, gas stations, hotels, movie theaters and attractions, which basically includes the location, address, phone number, and directions to a particular place. In addition it also provides information about a place of interest given by the user in the same location or anywhere within the United States. Additional features of the application include multiple views of maps, calling the place of interest with a single touch, navigating to the URL of the website of the place, calculating the distance to the place from the current or specified location and also checking traffic conditions. On the whole, it is a single integrated application that serves as a complete information repository on a smart phone.

1.2 Scope of Project

The scope of this project is to develop an Android smart phone based mobile application that is capable of delivering reliable and instantaneous location based information about pre-listed place types such as coffee shops, restaurants, fast foods, gas stations, hotels, movie theaters and also perform additional searches about a different place of interest in the current or different location. It is a GPS based android mobile application that senses the user's current location based on the latitude and longitude and provides precise and accurate location based information about a place of interest within the United States.

It is intended to provide a rich set of information about a particular place of interest through web services that includes

- Map View
- Address and Rating
- One touch phone call
- Navigate to the URL
- Driving Directions
- Distance from the current location
- Traffic information

The main feature of this application is that the information can be retrieved momentarily irrespective of the place and time. Since it is a mobile application, it becomes accessible while the user is at home or away and even when the user is moving around.

1.3 Significance of Project

Being a mobile application specifically developed for Android, it contains information about a particular place within any city in the United States, that is available to the user anywhere and at any time. Since it is developed on a smart phone that has advanced computing capabilities, it also lets the user interact with the application. There is no Android application that has been designed specifically to give complete and detailed information about a particular place of interest. This is mainly a geographical information serving application with an attractive UI. It provides the user with attractive one touch buttons for retrieving the desired information about a place. Instead of accessing several websites or several applications for each of their needs, this integrated application provides one stop information about attractions, hotels, restaurants, fast foods, coffee shops, gas stations, distance between two locations, and

directions from the current location. There are two compatible android based mobile applications that offers location based services, but are not elaborate and confined in their own ways. Google has a location based application known as Places [2] that renders information about a place of interest around the current location where the user has the hand held android based smart phone. But it does not provide information about places of interest in a different location. Poynt [3] is a mobile application that offers location based services about businesses, movie show times, restaurants, and also serves as local yellow pages. The main feature that differentiates this application from the other two are that in addition to providing location based information about places in the current location, it also provides the same type of rich information about any place in the United States that the user desires. It also has the capability of making one-touch phone calls and navigating to the URL of the website of the place. These features make this application wholesome and different from the others. With this one application, a person can explore a place, make or change plans momentarily or ahead of time, retrieve all sorts of information by accessing one single application.

1.4 Limitations of Project

Despite the smartness of the phone and the richness of the application, this project also has some limitations which could be both technical and data related. Since the core functionalities of the application are totally dependent on the web services provided by different hosts that maintain and operate the services, that runs on a dynamic and unpredictable environment, there are chances where there could be situations where the information may be partially or totally unavailable or may take unusually longer time to retrieve the information which could result in performance degradation.

The other major factor that could potentially hurdle the service could be lack of proper data management and maintenance by the web services. For instance, the database on the central server may be corrupted or out dated which might deliver wrong information to the users. So the performance, availability, and the reliability of the application may be altered by the hosting web services.

1.5 Definitions, Acronyms, and Abbreviations

The following terms and definitions have been used in the project.

Android - Android is a mobile operating system that consists of a software stack of Java applications running on a Java based object - oriented application framework on top of Java core libraries running on Dalvik virtual machine that features JIT-Just in time compilation. [4]

SDK - Software Development Kit is typically a set of development tools that allows for the creation of applications for a certain software package or software framework. [5]

Android SDK - Android Software Development Kit is a software stack for mobile devices that includes an operating system, middle ware, and key applications. It provides the tools and APIs necessary to develop applications on the Android platform using the Java programming language. Android includes a set of core libraries that provides most of the functionality available in the core libraries of Java programming language. Every Android application runs its own process, with its own instance of the Dalvik virtual machine. [6]

ADT - Android offers a custom plug-in for the Eclipse IDE, called Android Development Tools that is designed to give a powerful, integrated environment in which to

build Android applications.[10] It extends the capabilities of Eclipse to quickly set up new Android projects, create an application UI, debug the applications using the Android SDK tools, and even export signed APKs in order to deploy your application on a real device. In general, developing in Eclipse with ADT is a highly recommended approach and is the fastest way to get started with Android. [10]

Eclipse - Eclipse is a multi-language software development environment comprising an integrated development environment and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and in other few other programming languages with their corresponding plug-ins. [11]

Smart Phone - A smart phone is a mobile phone that offers more advanced computing ability and connectivity than a contemporary basic feature phone. Smart phones and feature phones may be thought of as handheld computers integrated within a mobile telephone. A smart phone usually allows the user to install and run more advanced applications. Smart phones run

complete operating system software providing a platform for application developers. [1]

Java - Java is a general-purpose, concurrent, object-oriented programming language.[22]

Object Oriented Programming (OOP) - Object Oriented Programming is a type of computer programming paradigm that uses objects which are data structures consisting of data fields and methods together with their interactions to design applications and computer programs. The programming techniques may include features such as encapsulation, abstraction, modularity, polymorphism, and inheritance. [8]

Dalvik Virtual machine - It is the virtual machine used in Google's android operating system. Dalvik is therefore an integral part of Android, which is typically used on mobile devices. Every android application is converted into the Dalvik Executable before it is executed. [7]

GIMP - GIMP stands for GNU Image manipulation program it is a free software used for image retouching and editing. It can be used a s a simple paint program or as an expert quality image providing software. It is available as a free source software and runs on most operating systems. [16]

HTTP - Hypertext transfer protocols is a networking protocol used for data communication in the world wide web. It defines how messages are formatted and transmitted and what actions web servers should take in response to the http requests they receive. [13]

Dia - Dia is a free and open source general purpose software used for drawing diagrams. It has a modular design with several shape packages such as flow charts, network diagrams, entity-relationship diagrams etc.. [14]

UML - Unified Modeling Language is a standardized general purpose modeling language used in the field of software engineering. It includes a set of graphic notation techniques to create visual models of software processes throughout the software development life cycle. [15]

DOM Parser - An XML parser converts an XML document into an XML DOM object. DOM (Document Object Model) is an interface-oriented Application Programming Interface that allows for navigation of the entire XML document as if it were a tree of node objects representing the document's contents. [18]

XML - Extensible Markup Language (XML) is a set of rules for encoding documents in machine-readable form. [23]

API - Application Programming Interface defines specifications specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. [24]

Google Maps API - Google Maps is a web mapping service application and technology provided by Google, free for non-commercial use, that powers many map-based services. It offers street maps, a route planner for traveling by foot, car, or public transport and an urban business locator for numerous countries around the world. [25]

Google Directions API - It is a service that calculates directions between locations using a HTTP request. Directions may specify origins and destinations either as text strings or as latitude/longitude coordinates. The Directions API can return multi-part directions using a series of waypoints. This service is generally designed for calculating directions for static addresses for placement of application content on a map. [17]

Yahoo local search API - Yahoo local search API is used for performing searches using http requests and the response is a result of the query specified with the

http request, which is usually the URL with a search criteria specified. The response generated is usually an xml document, but could also be obtained in other formats such as JSON (Java Script Object Notation).

CHAPTER TWO

SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Overall Description

2.1.1 Project Design

Having chosen to design and deploy the application on the Eclipse IDE, the Android Development Tools (ADT) plug-in adds powerful extensions for the Eclipse IDE which helps creating and debugging Android applications much easier and faster [6]. The project was implemented in three phases and then testing and debugging was done. The final developed package was then deployed on a real device in debuggable mode.

The first phase consisted of analyzing the project requirements and finalizing both the functional and non-functional aspects of the project.

2.1.2 Project Perspective

GoBuddy is capable of providing rich set of information about a place of interest in the current or desired location. The main functionality of the project depends on the various web services hosted by Google and Yahoo.

2.1.3 System Interface

The client or the end user will use a smart phone to access the application. Being specifically designed for Android, the user's device must be an Android based mobile device.

2.1.4 Software Interface

GoBuddy has been designed for use by the general audience. It will be capable of functioning on an android based mobile device. The software chosen to develop the application includes Java, XML, Google Maps API, Google Directions API, Yahoo Place Finder API, and Yahoo local search API. The development environment was Eclipse IDE. And the output module was the Android Emulator which is a simulated android phone that works on Windows.

2.1.5 Communication Interface

The communication interface is any smart phone that is capable of running on Android or an Android emulator window.

2.1.6 Hardware Interface

To run GoBuddy on a smart phone, it should be an Android based mobile device with the GPS and Wi-Fi turned on. GoBuddy will run on any T-mobile or Samsung based android phone.

2.1.7 Memory Requirement

Being a mobile phone based project there is a restriction to the memory for each application being developed. GoBuddy as of now consumes 3MB on a mobile phone which meets the limitation criteria set by Google for such applications.

2.2 Functional Requirements

2.2.1 Core Functionalities of the Project

GoBuddy serves as an information repository and provides the users with information on any predefined common place types in and around the current location. The user can also search for his own place of interest in the current or any desired location within the United States. The information provided about each place includes displaying a map view pointing to the current location and the place of interest, displaying the address, phone number and rating of the place if any, making a phone call to the place with a single touch, navigate to the URL of the place without having to explicitly load it on a browser window, calculate distance to a point from the current location and also get driving directions. All these features makes this as one wholesome application

with all the features integrated in a very user friendly manner.

2.2.2 Use Cases

2.2.2.1 User-Request. The user touches the pre-defined icons on the mobile device which then displays a map view, address and directions about the place of interest in a tabbed view. The search results returned are stored in global arrays and retrieved one at a time and displayed on the map. This makes the information very precise and less confusing to the user.

2.2.2.2 User - Search. The user touches the Find button and then is allowed two choices, whether to perform the search in the current location or in a different location. Then the information is retrieved based on the user's choice and a list of map view, addresses, and directions are displayed in a tabbed view.

There are the two main modules of functionalities

Common places of interest

- Coffee Shops
- Restaurants
- Fast foods
- Gas Stations
- Hotels

- Movie Theaters
- Attractions

Search for user's choice of place of interest

- Current location
- Desired location

In each module for each criteria based on the user's choice, the following information is being rendered to the user.

- Map
  - Displays a Map view with the current location and the location of the place of interest.
  - Satellite View
  - Traffic Conditions
  - List of the 10 closest places
- Address
  - Name of the place
  - Address of the place
  - Phone number
  - Rating
  - Call Button to make a phone call with a single touch

- URL Button to navigate to the URL of the website of the place of interest.

- Directions
  - Display the driving directions from the current location to the specified location on the map.
  - Display the distance between the two points on the map.

2.2.2.3 Assumptions and Dependencies. The core functionalities of the project mainly depend on the services rendered by the Web servers managed by Yahoo and Google.

CHAPTER THREE

DESIGN

GoBuddy, being an android application, has a framework which is built and organized around the common Model-View-Controller pattern. The framework provides tools for constructing a Controller that handles key presses and taps, and a view that gives the graphical representation to the user's screen.

3.1 Basic Concepts of Android and Model/View/Controller (MVC)

The view which displays the graphical part of the Android's interface frame work to the user, has been implemented as a tree of subclasses of the View class. This represents a rectangular area on the screen and the root of this tree is the main application window. And each external action by the user which could be a key press or a tap is implemented as an event queue and each event gets a response from the controller [20]. The controller acts as a intermediary between the model and the view and processes the http requests to the web APIs.

3.2 Architecture of GoBuddy

Being an application developed for the general audience, the application's UI component was designed with

great care, in an effort to make it very simple and user friendly for the end user. The main application screen was composed of a grid of icons, which on a single touch provides a handful of information about a place without even having to maneuver the device much. When the user touches an icon, a http request is sent to the web servers and the response element is a XML file, which is then parsed using a DOM (Document Object Model) parser and the required input fields are extracted and displayed on the screen. The user has the provision of easily navigating between the screens with the help of buttons.

3.3 User Interface Design

The user interface plays a very important role in making the application attractive and popular. The screen views and the navigation between various screens were designed in a way to provide the user with maximum ease to use the application.

3.4 Structure

The application basically serves the request of the user on a single touch or tap on the screen. The basic view has two main modules. The first component consists of the predefined icons, that displays the information with a single touch and the second module consists of the search

option provided to the user, where the user is prompted to enter his place of interest in the same or different location.

Main Screen → Touch Icons → Output Screen

Main Screen →Search Screen→ Output Screen

3.4.1 Module 1

The application comes with a set of predefined search icons for some commonplace types such as coffee shops, restaurants, fast foods, gas stations, hotels, movie theaters, and attractions. The application displays a set of icons for all these components on the main screen, which when touched or tapped by the user makes a http request to the web servers and displays the response on the output screen. The output screen has three main components, which includes the map, address, and the directions. Each of these components has their own options and features that render a very rich set of information to the user about the particular place.

3.4.2 Module 2

The application also provides a search icon on the main screen, which when touched or tapped navigates to the search screen. Here the user enters the search criteria based on his interest. If it is the current location then only the place of interest is entered. If it is a

different location, then the name of the city and the state should be given. Then the application performs a http request to the web servers and displays the output in a similar manner like the default search results.

3.4.2.1 Use Case Diagram. The following use case diagram shows the main functionalities that have been defined for the application.

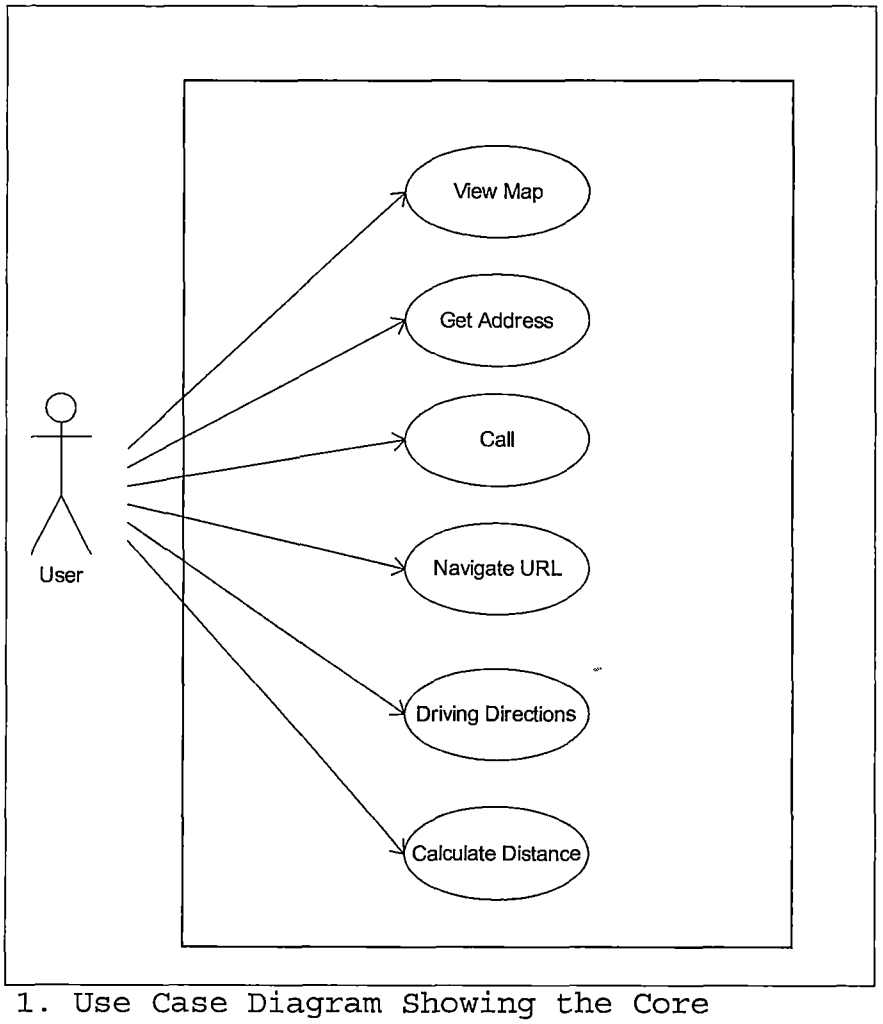


Figure 1. Use Case Diagram Showing the Core Functionalities of the Project

3.5 Screen Descriptions

Based on the requirements analysis done so far and on the use case diagram designed the following screens and views were proposed to be featured in the application.

3.5.1 Main Screen

This is the screen which is the basic view available for interaction with the user. It has a grid view with a set of icons, which generates an event when touched or tapped.

Table 1. Main Screen

Icons	Functions
Coffee	Displays a list of coffee shops that are in close proximity to the current latitude and longitude.
Restaurants	Displays a list of restaurants shops that are in close proximity to the current latitude and longitude.
Fast Foods	Displays a list of fast foods that are in close proximity to the current latitude and longitude.
Gas Stations	Displays a list of gas stations that are in close proximity to the current latitude and longitude.
Hotels	Displays a list of Hotels that are in close proximity to the current latitude and longitude.
Movie Theaters	Displays a list of movie theaters that are in close proximity to the current latitude and longitude.
Attractions	Displays a list of attractions that are in close proximity to the current latitude and longitude.
Search	Navigates to a search screen where the user has a choice of performing his place of interest either in the current location or a different location.

3.5.2 Search Screen

When the user clicks the search icon, the screen is navigated to another screen, which gives the user the choice of searching in the current location or a different location within the United States. The screen has buttons and text area where the user can enter his place of interest, name of the city and a drop down list for selecting the state.

Table 2. Search Screen

Buttons	Functions
Current Location	Displays a text area to enter the user's place of interest.
Different Location	Displays text editors for entering the place of interest and the name of the city and a drop down list for selecting the state.
Search	Initiates the search query and displays a list of the search criteria in the current location or different location based on the user's choice.
Reset	Clears the text editors.
Back	Navigates the screen back to the main screen.

3.5.3 Table for Output Screen

The output screen consists of a tabbed view with three tab areas - Map, Address, Directions. Each of the tabs has its own perspective and options.

3.5.4 Map View

Displays a map view with markers pointing to the current location of the user, and the location of the place of interest. The map view also has buttons that provides the user with optional views. This is the main view for navigating through the result set. The address view and the directions view display information with respect to the current item that is being displayed in the Map View.

Table 3. Map View

Buttons	Functions
Previous	Displays the map view with markers pointing to the current location of the user and the previous item in the list returned by the search query, if the current view is not displaying the first item in the list.
Next	Displays the map view with markers pointing to the current location of the user and the next item in the list returned by the search query, if the current view is not displaying the last item in the list.
Traffic	Displays the map view with the current traffic conditions on the freeways.
Satellite	Displays a satellite view of the map.
Street	Displays the street view of the map.
Zoom Control	Displays the buttons for zooming the map in and out.
Back	Navigates to the previous screen that was in view.

3.5.5 Address View

The address view is a text view that displays the name of the place, city, state, phone number, and the rating. It also has some buttons that provides the user with few options.

Table 4. Address View

Buttons	Functions
Call	Initiates a phone call to the particular place when a touch or tap event is triggered.
URL	Navigates to a browser window that displays the website of the selected place of interest.
Back	Navigates to the main screen.

3.5.6 Directions

The directions view is a text view that displays the directions to a particular place from the current location. It also displays the distance between the two points on the map.

Table 5. Directions View

Buttons	Functions
Back	Navigates to the main screen.

3.6 Object Model Diagrams

The application was designed to have the following classes based on the requirements analysis and the functionalities that have been defined. This represents the objects required to build the application and the relationships between the various components.

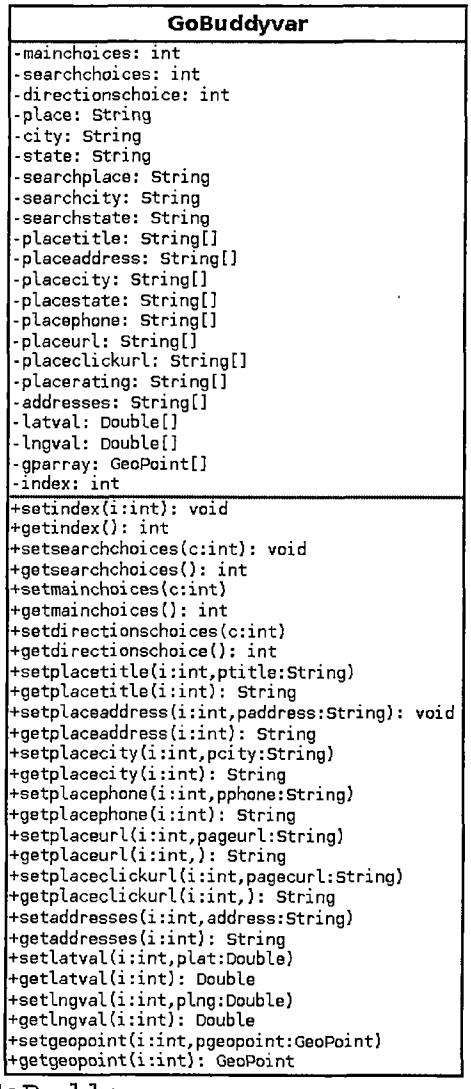


Figure 2. Class GoBuddyvar

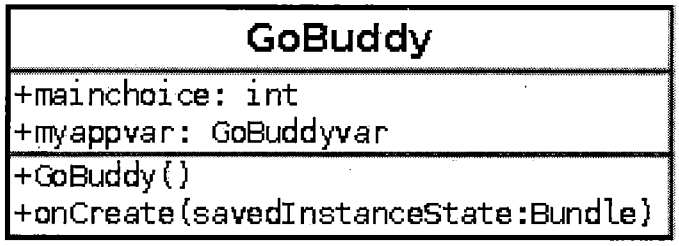


Figure 3. Class Gobuddy

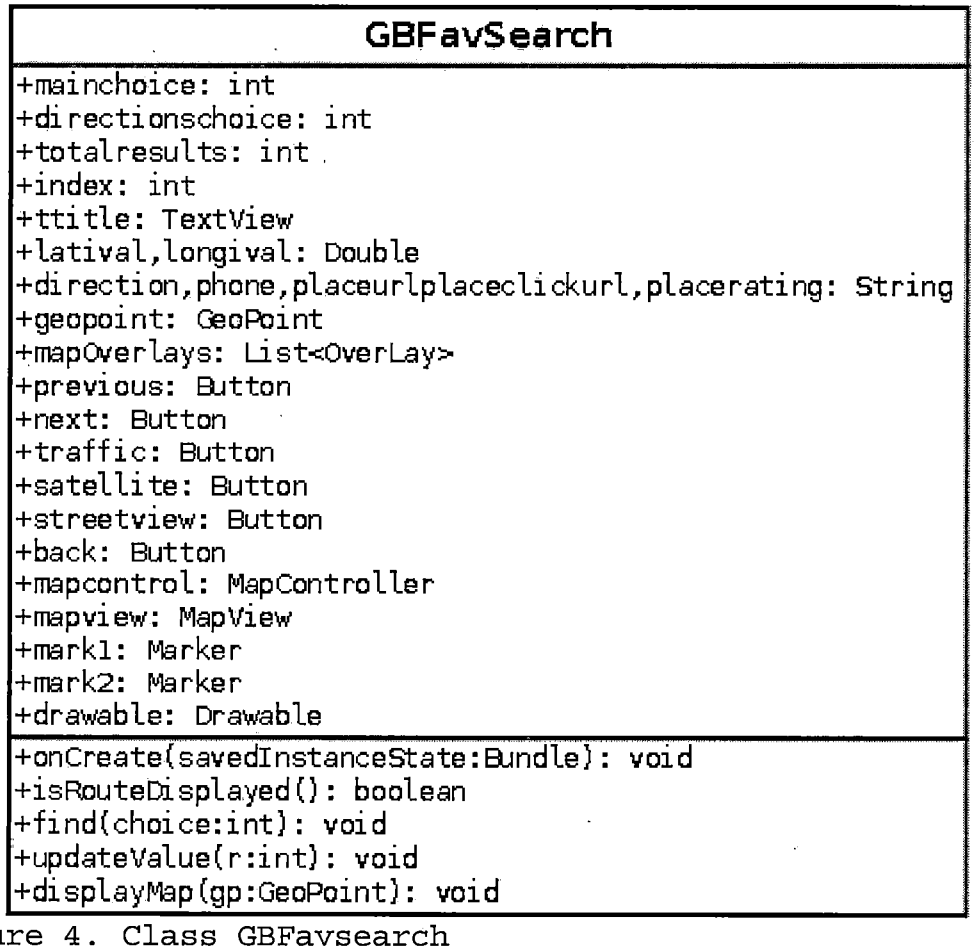


Figure 4. Class GBFavsearch

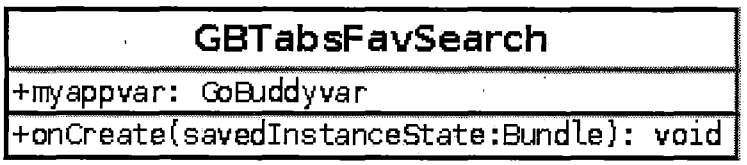


Figure 5. GBTabsFavSearch

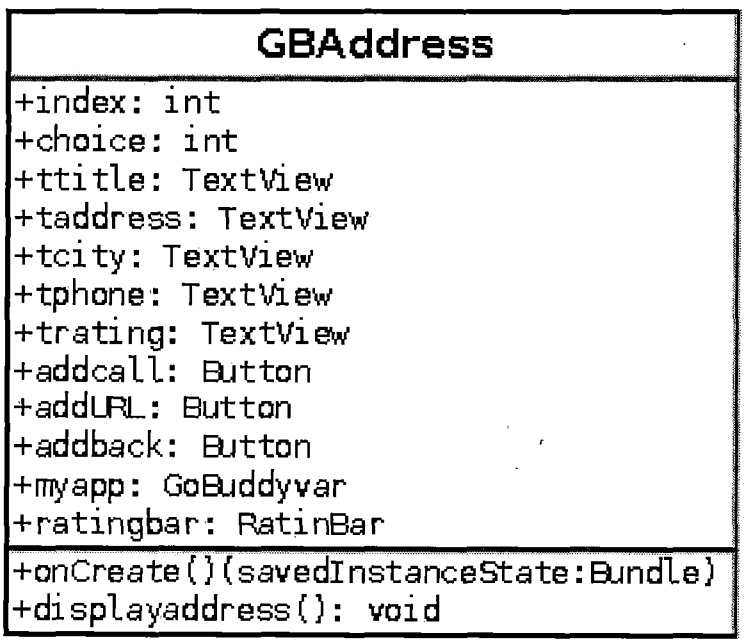


Figure 6. Class GBAddress

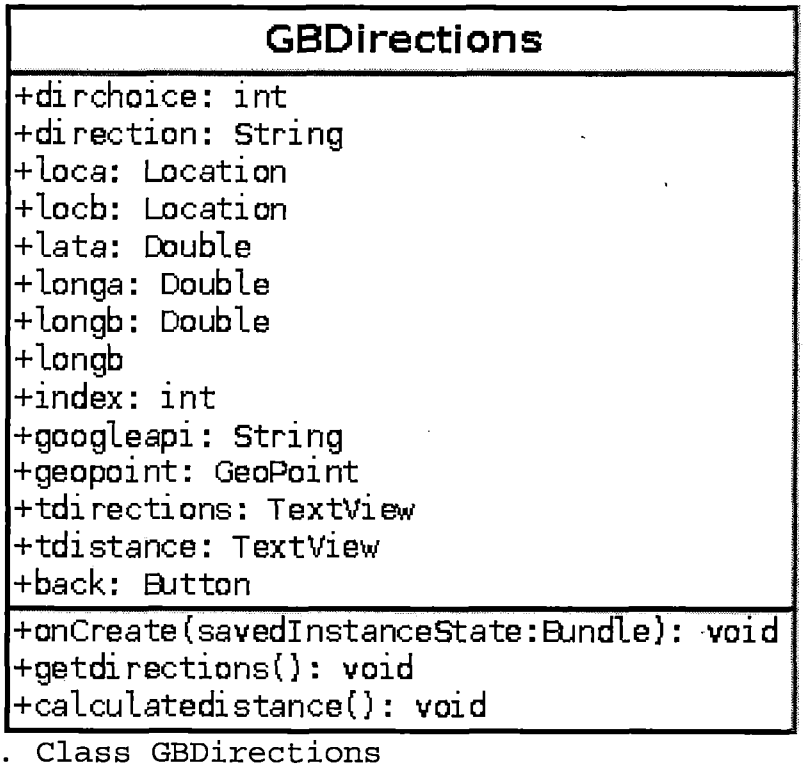


Figure 7. Class GBDirections

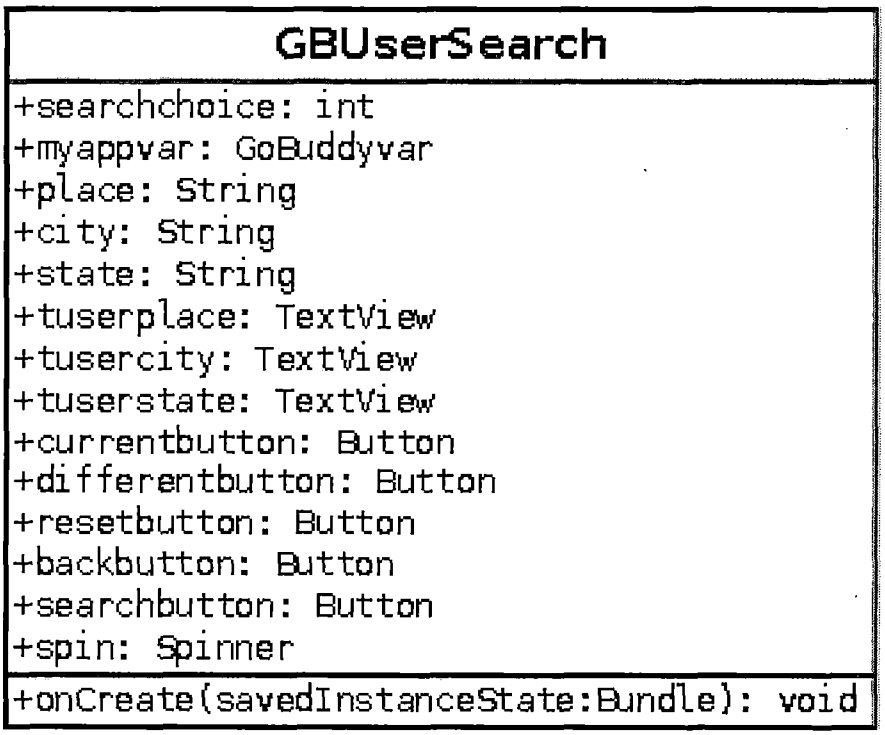


Figure 8. Class GBUserSearch

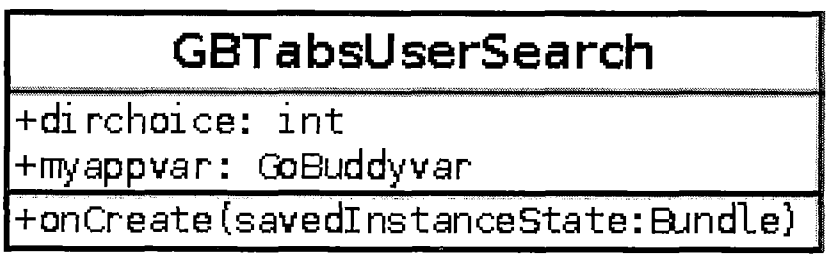


Figure 9. Class GBTabsUserSearch

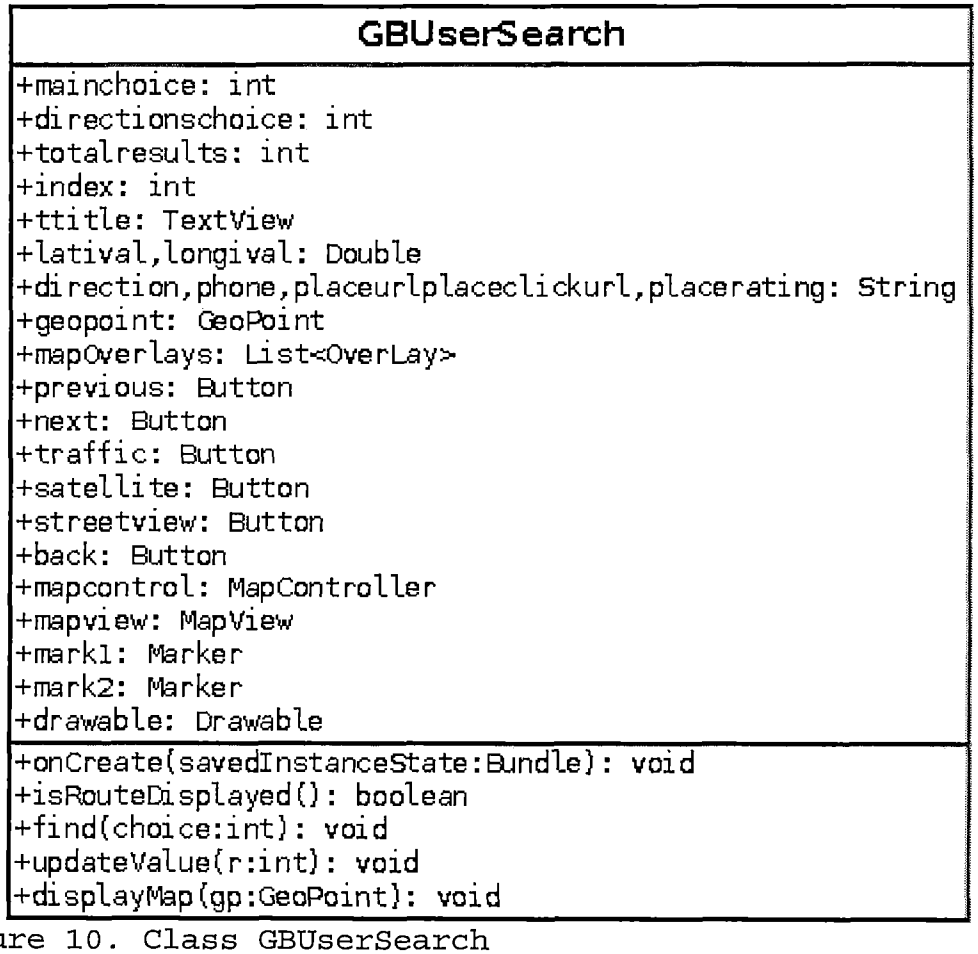


Figure 10. Class GBUserSearch

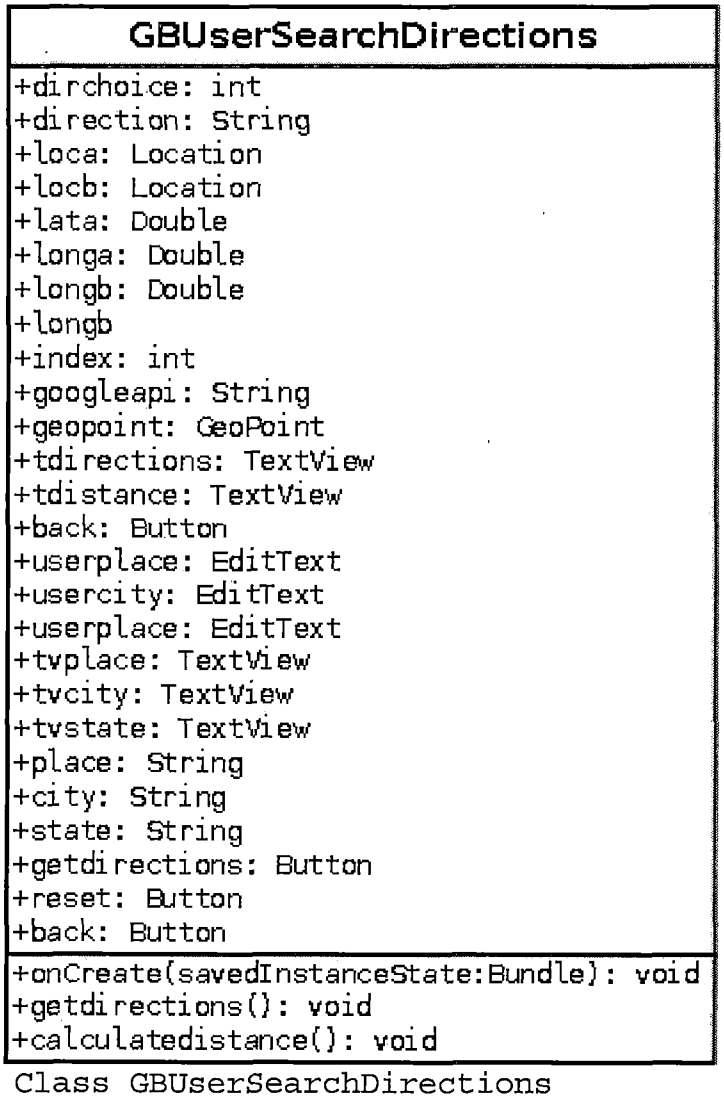


Figure 11. Class GBUserSearchDirections

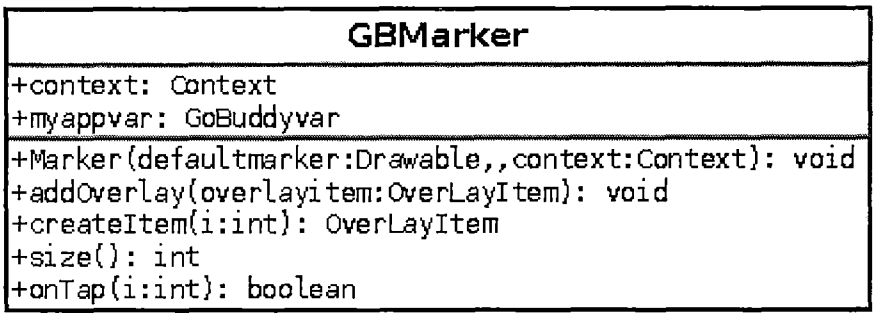


Figure 12. Class GBMarker

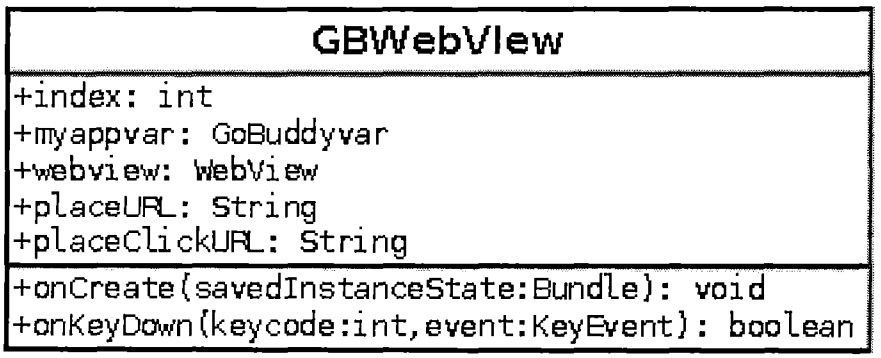


Figure 13. Class GBWebView

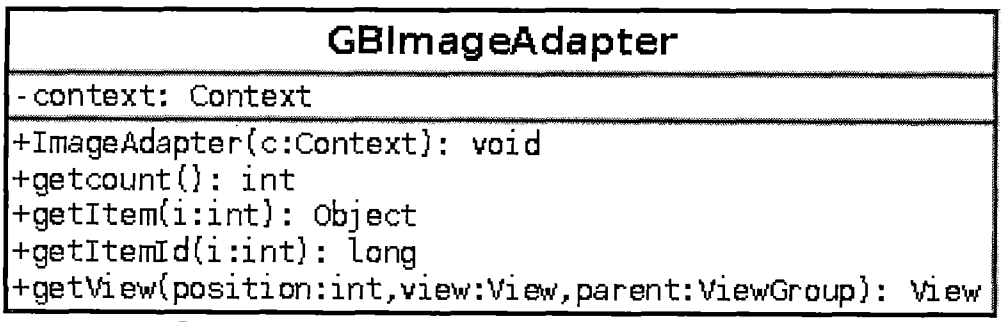


Figure 14. Class GBImageAdapter

3.7 Relationship Modeling

The following diagrams illustrate the relationships between the classes that have been used in this application.

3.7.1 Basic Search Module

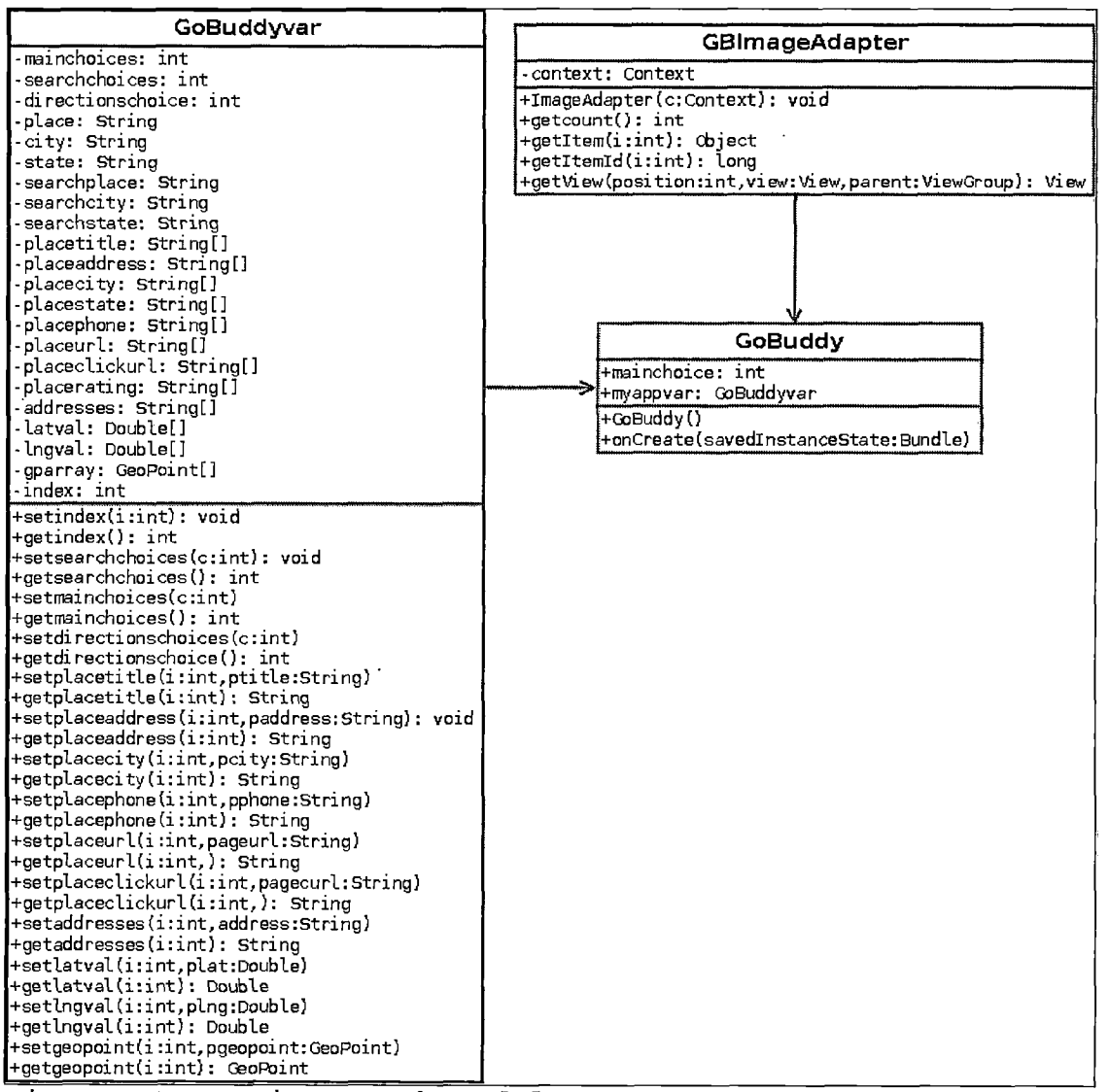


Figure 15. Basic Search Module

3.7.2 Pre-Defined Place Finder Module

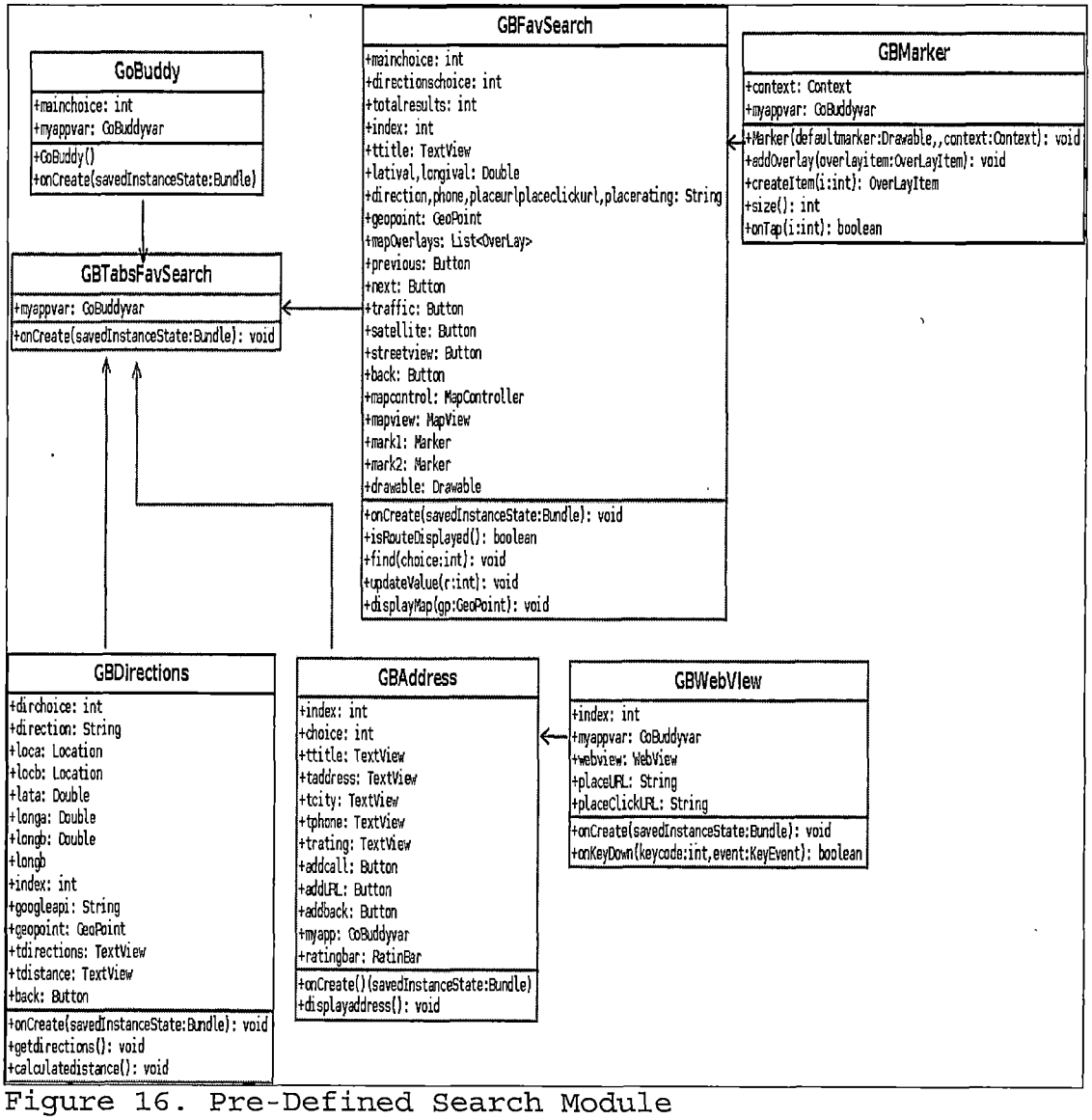


Figure 16. Pre-Defined Search Module

3.7.3 User Search Module

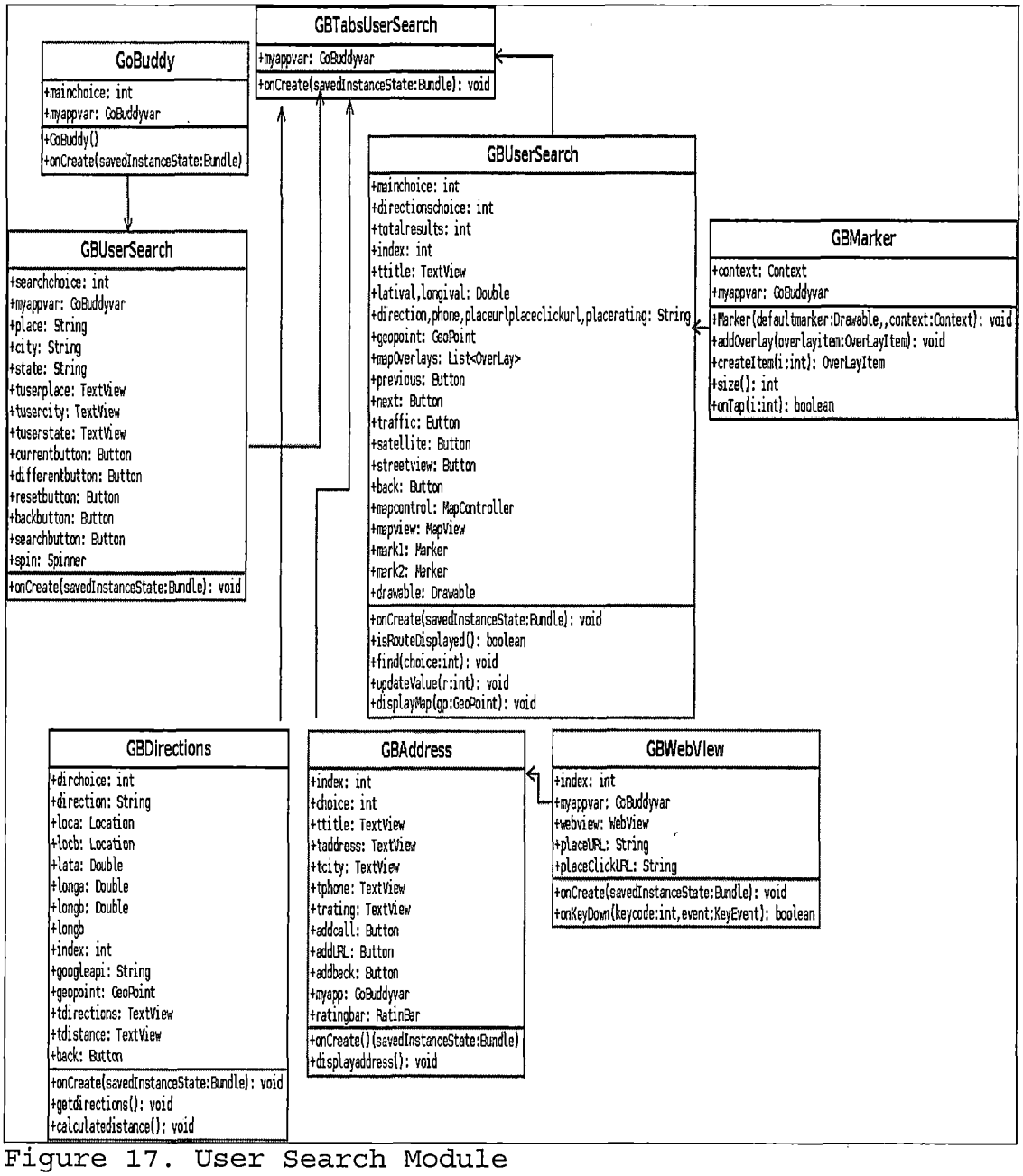


Figure 17. User Search Module

CHAPTER FOUR

IMPLEMENTATION

In software development, implementation of the code is the most important phase in which the requirements collected during the requirements analysis, and designed using the design tools are put together and the actual code is written to achieve the desired software.

4.1 Basic Concepts of Android Frame Work

Android is a mobile operating system that was initially developed by Android Inc. and then bought by Google. [4]. It basically consists of a software stack that comprises of Java applications running on a Java-based, object-oriented application framework on top of Java core libraries running on a Dalvik virtual machine that features JIT (Just In Time) compilation. [4]

Android applications are developed mostly on Eclipse IDE (Integrated Development Environment) using the Java programming language, even though few other IDE are also available. The application consists of three main categories of files which includes the java files, XML files which has the UI design and the Android Manifest file which contains the essential information that is given to the Android system before any application code

can be run. The compiled Java code and the other resource files required by the application are packed into one single android package which is an archived file with .apk extension. This single file is the one that is installed and run on the emulator or on mobile devices. All the files that are bundled to form a single .apk file is one whole application. Each android application has its own virtual machine and each one runs its own Linux process. [6]

Even though it is a mobile application being developed for an Android mobile device, the main application development is done on a personal computer on an appropriate development environment with suitable development tools and languages. The output is viewed and tested on an Android emulator and the final package file, which has all the files and resources bundled into one single .apk file is deployed on the real device. The android emulator is a virtual mobile device running on a computer that mimics all the hardware and software features of a real mobile device, except that it cannot place or receive phone calls.

4.2 Implementation Outline of GoBuddy in Android

GoBuddy was developed on Eclipse using Java and XML. The user interface was implemented using XML files that defined the layouts and the views of the components. These views were accessed through the class files written in Java.

The most important component of an Android project is the application which is the root. Each application may have one or more activity components. Each activity is responsible for displaying the visual user interface to the user for interaction and capable of starting another activity. This forms the main underlying concept of GoBuddy. The entire project is organized into various activities, and each activity corresponds to a view whose layout component is set as content view from the corresponding XML files which has the declared layouts and components. The activities are activated by asynchronous messages called Intents. The intents are intent objects that holds the content of the service. [20]. It is responsible for deciding what the activity is intended for such as making a call, act as launcher etc.

The Android XML files were designed to hold different layouts, components and widgets are set as content views when each activity is started.

The android manifest file is the most important file that contains the core information for the application code to run. The components and activities are declared in the Android manifest file that is being bundled to the package. The various activities of the application are declared in this file, only after which an activity can be activated using the intent objects. In addition the external libraries that are consumed by the application and the class file that maintains the values parallel throughout the application are declared with the manifest file. Also, permissions that are required for the application to run such as internet connect and the provision to access the latitude and longitude are also specified. This is very important since the application requires that the GPS and the Wi-Fi be turned on in the device, and for the GPS and Wi-Fi to function properly these permissions must be declared in the manifest file.

```
<uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <application android:name=".GoBuddyvar" android:debuggable="true"
    android:icon="@drawable/icon" android:label="@string/app_name">
    <uses-library android:name="com.google.android.maps"/>
        <activity android:name=".gobuddy"
                android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".GBUserSearchDirections"
                android:label="@string/app_name" />
        <activity android:name=".GBWebView"
                android:label="@string/app_name" />
```

Figure 18. Code Snippet for the Android Manifest File

In addition to all of these, there are resource files in which the values of strings, and images can be saved. These values can then be retrieved by making a reference to the resources.

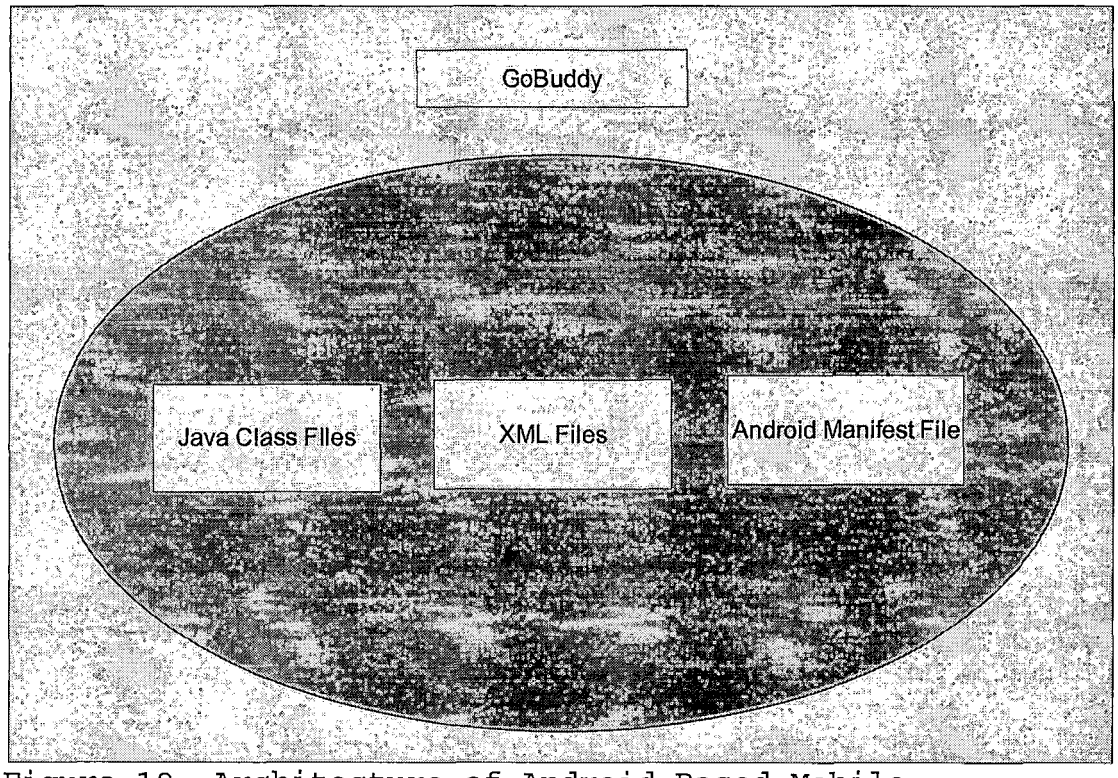


Figure 19. Architecture of Android Based Mobile Application

4.3 Interaction of GoBuddy with the Web Services

A web service is a software system that facilitates communication between electronic devices over a network. The interaction with the web services is accomplished by the description specified by SOAP (Simple Object Access Protocol). HTTP (Hyper Text Transfer Protocol) is used for message transmission and XML (Extensible Markup Language) is used as the message format. Web APIs are advancements of web services that define the methods to invoke http requests and also defines the format of the responses. The

http requests are processed by the web APIs and the responses are usually in XML (Extensible Markup Language) or JSON (Java Script Object Notation). GoBuddy uses web APIs hosted by Google and Yahoo.

There are several methods used to access the web APIs. The method used in this project is REST (Representational State Transfer), which defines the HTTP methods such as GET, POST etc., and enables interaction with stateful resources. HTTP is a request-response protocol, that falls into the client-server model. The client is the application trying to access the information and the server is the application running on the host side.

The HTTP request method used with this project is HTTPGet, which retrieves the information in the form of an entity, that is requested by the URL. The services take the http requests, executes them in their server and returns the http responses to the calling interface. The HTTP response is the result of the query request which is processed on the server side. There are several formats of retrieving this information from the hosting servers, the most common ones being XML and JSON. As it has been described earlier, the application's services are mainly Web APIs hosted by Google and Yahoo.

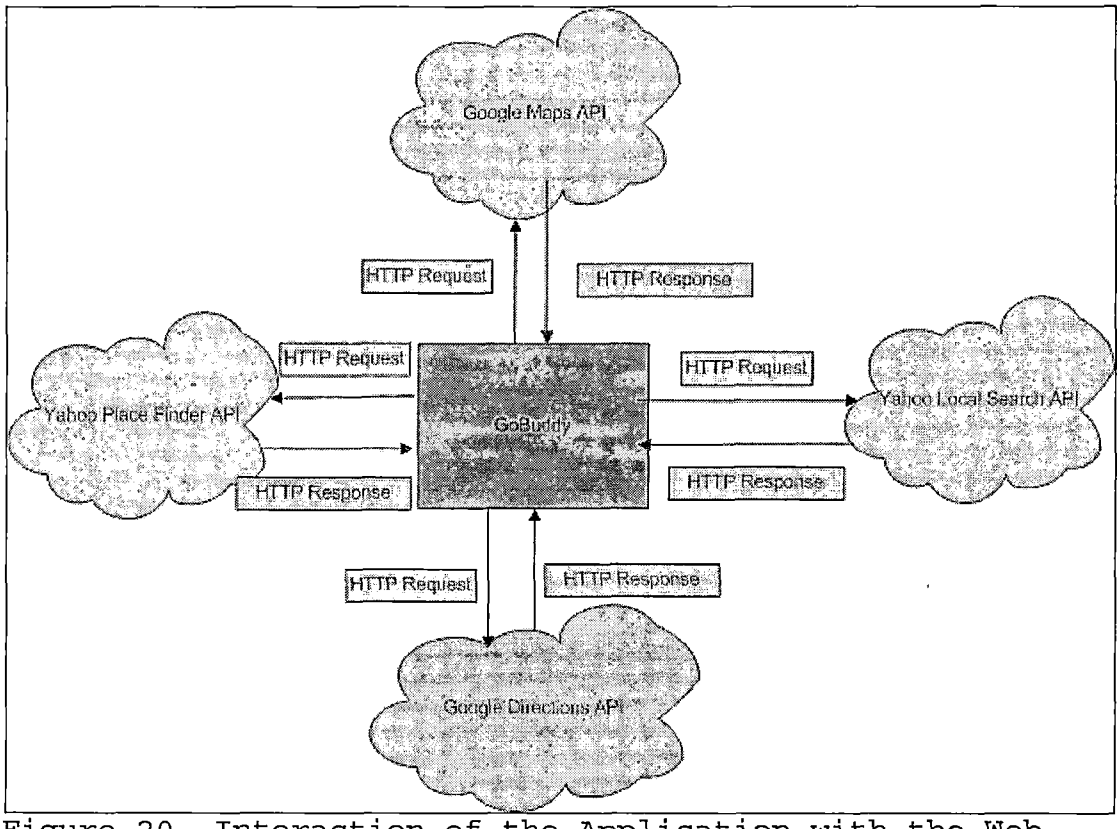


Figure 20. Interaction of the Application with the Web Services

GoBuddy makes use of four different APIs to deliver the service that is has been designed for. The basic information retrieval is done through the Yahoo Local Search API which takes the latitude and longitude as input parameters in addition to the search query such as "hotels", "gas+stations" etc. The latitude and longitude are obtained from the in-built GPS on the smart phones. The search term is either provided in the pre-listed request mode based on the user's selection or the text

retrieved from the user's choice. The results obtained from the Yahoo Local Search API forms the base on which the entire application is built on and are utilized by the other API's correspondingly.

Yahoo Place Finder API is used for geocoding and reverse geocoding. The latitude and longitude got from the GPS are transformed into textual street address. And the starting address provided by the user in case the user is searching for a place in a different allocation, the address is converted to the corresponding altitude and longitude. These transformations are used for getting the driving directions and distance between two points.

Google Maps API is used for displaying the map and placing the markers as overlays on top of the maps corresponding to the user's current location and the location of the place of interest. The latitude and longitude values obtained from the Yahoo Local search API are utilized for this.

Google Directions API is used for finding the directions and distance between two locations. The values need to perform this service is retrieved from the Yahoo Local Search API and the conversion of the latitude and longitude to street address and the vice versa are obtained from the Yahoo Place Finder API.

The results obtained from the search query in all of the four web services used in developing this application are then parsed using a DOM (Document Object Model) parser which extracts an instance of the http response as a document. Then, the document elements are retrieved like that of tree with the outermost tag in the XML file being the root and the inner tags being the NodeList and Nodes. The values of the nodes are then stored in global arrays that are declared in a class separate from the other classes that handle the various activities. The global arrays function as parallel arrays whose values are available and maintained throughout the life cycle of the application. This is a very important feature of the project that clearly demonstrates the object oriented concept of encapsulation.

```
public class GoBuddyvar extends Application{

        private String[] placerating=new String[20];
        private String[] addresses=new String[20];
        private Double[] latval=new Double[20];
        private Double[] lngval=new Double[20];

public void setplacerating(int i,String prate){
                placerating[i]=prate;
        }
        public String getplacerating(int i){
                return placerating[i];
        }
        public void setaddresses(int i,String address){
                addresses[i]=address;
        }
        public String getaddresses(int i){
                return addresses[i];
        }
        public void setlatval(int i,Double plat){
                latval[i]=plat;
        }
        public Double getlatval(int i){
                return latval[i];
        }
```

Figure 21. Code Snippet for Demonstrating Encapsulation using Global Array Declarations

The android emulator window, which is a simulated phone on the computer was used for viewing the output during the development and testing phases.

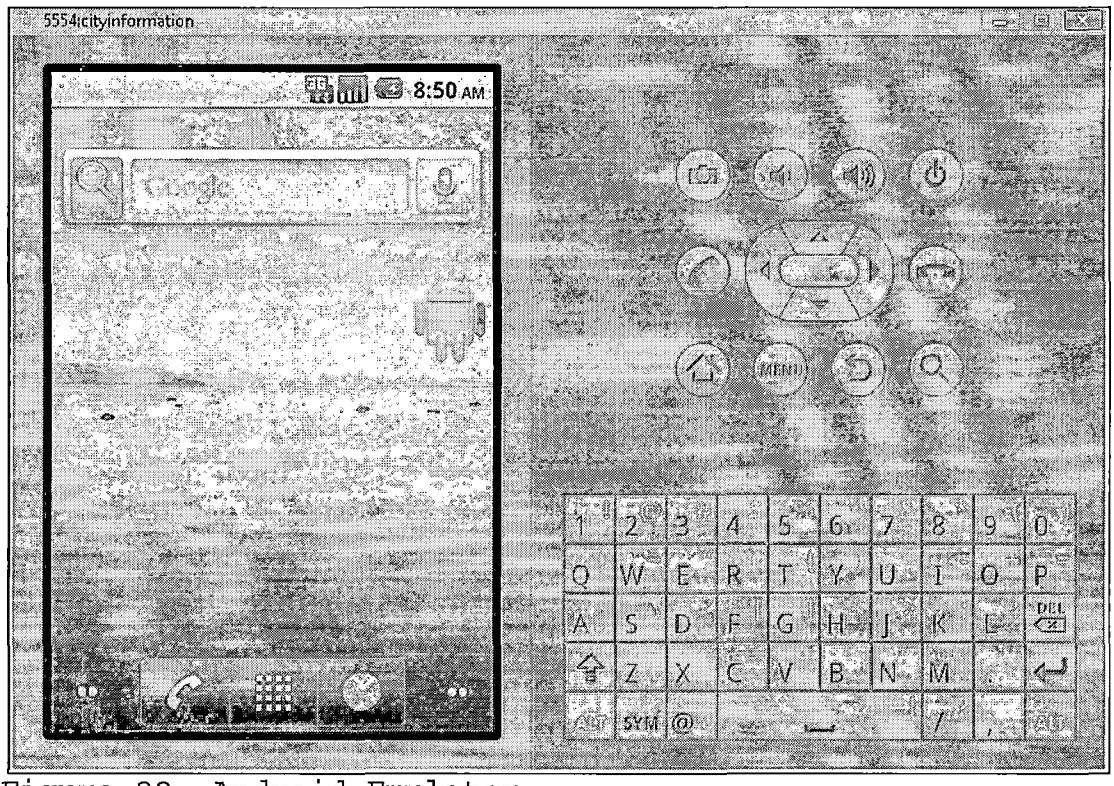


Figure 22. Android Emulator

4.4 Components and Controls

The user interface in android is built using the View and View Group classes. View is the base class for holding the other UI objects called widgets. For example TextView, EditText, ScrollView, Buttons are all widgets that can be placed on the base container. The View group classes are used for determining the layouts for the views such as grid layout, tabbed layouts etc. As mentioned earlier, the view and view group objects were declared in an XML file, which defined the layouts and the appearance and position of the UI components. A widget is a view object like a

button or zoom control, that acts as the interaction tool between the user and the application. The layouts used in this application are Linear Layout - where the components are placed in the order they are declared, Relative Layout - where the positions of the components are declared with relative to each other like toRightOf, align_parentTop etc.., Tabbed Layout - where each tab area has its own View, Grid Layout - where images are arranged in a grid fashion using an Image Adapter.

4.5 Functionalities

The main concept of the application is to initiate a location based query, depending on the interest of the user and display a list of the place of interest that are in close proximity to the current or different location.

4.6 Common Places of Interest Near the Current Location

The main screen is a grid view with a list of image icons, with each image corresponding to a place of interest such as Coffee Shops, Restaurants, Fast foods, Gas stations, Hotels, Movie theaters and a search icon which directs the view to start another activity to initiate user's search.

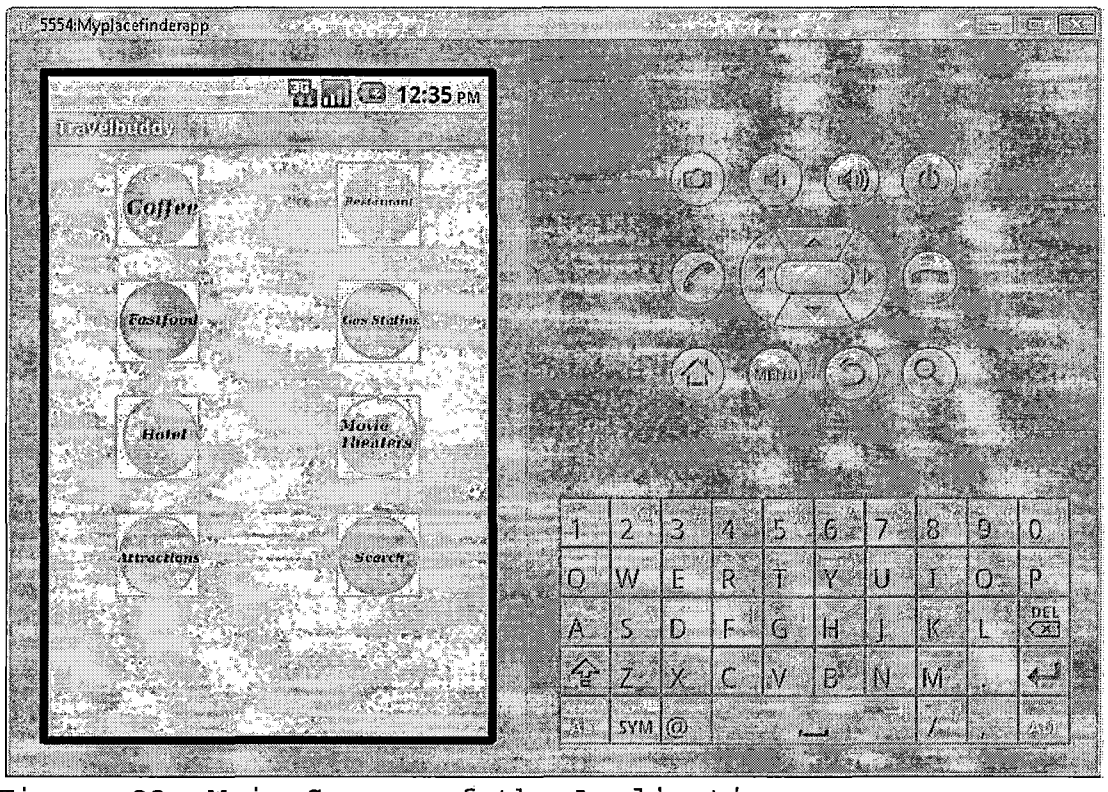


Figure 23. Main Screen of the Application

4.6.1 Coffee Icon

When the user clicks the coffee icon a http request call is made to Yahoo local search API and the query performs a search for Coffee Shops in and around the current latitude and longitude and the results returned is sorted by distance. The http response is an XML document, which is then parsed using a DOM (Document Object Model) parser. The results are then displayed in a tabbed view. The latitude and the longitude obtained from the query results are used to form the GeoPoint. A marker is created and placed on the user's current latitude and longitude

and on the latitude and longitude of the first item in the result set returned by the Yahoo local search API. Simultaneously, the name, address, and phone number of the corresponding result item is placed on the address view and the directions from the user's current location to the place is also displayed on the directions tab. The user gets all the information about a place of interest all at the same time. The user just has to click on the various tabs to get the information he is looking for. The number of results returned by the search is limited to 10.

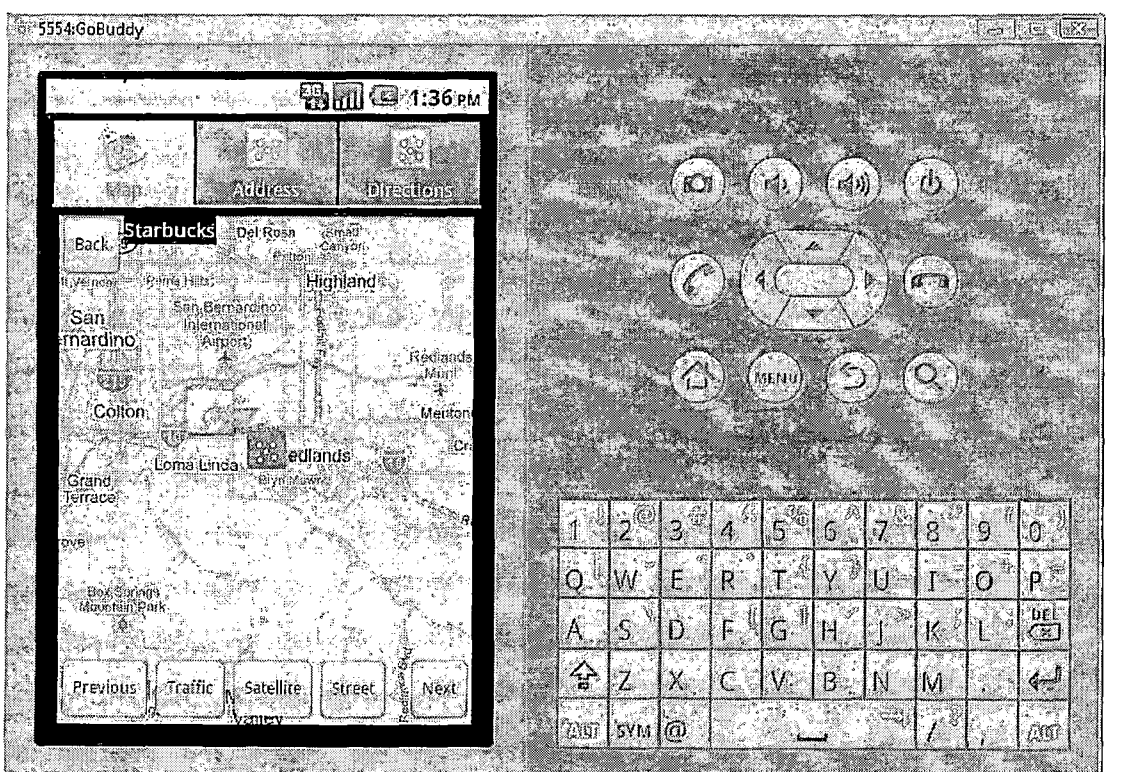


Figure 24. Map View when Coffee Icon on the Main Screen is Touched

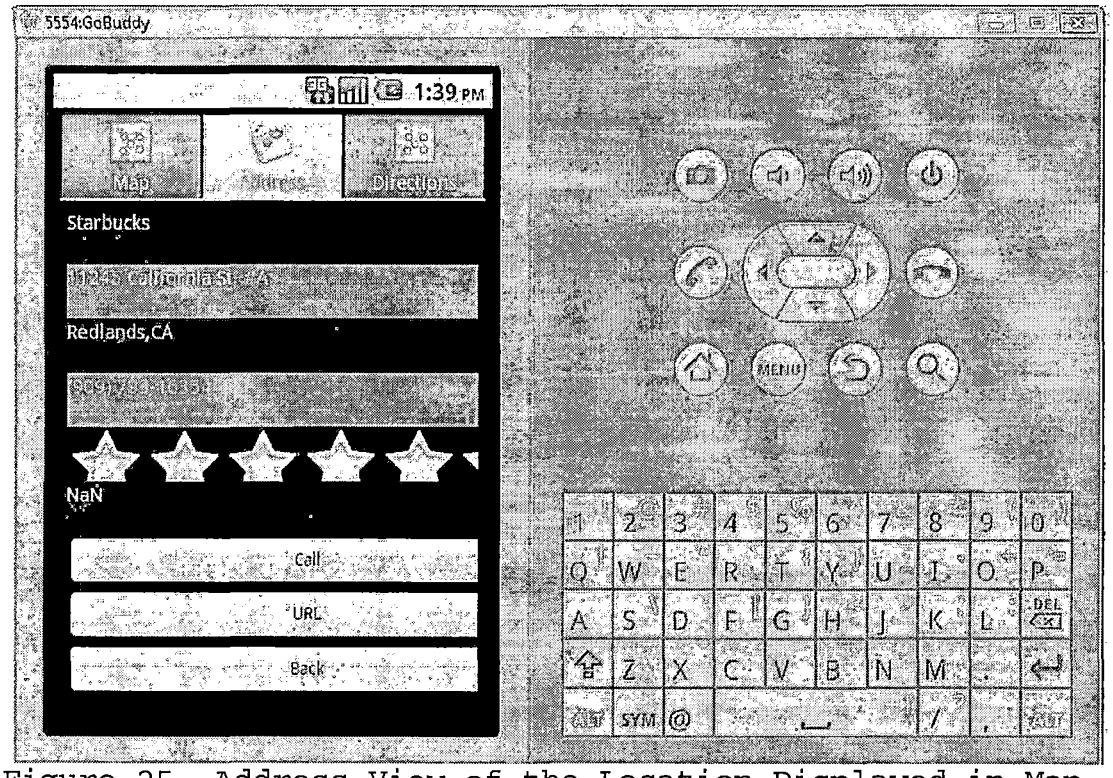


Figure 25. Address View of the Location Displayed in Map View

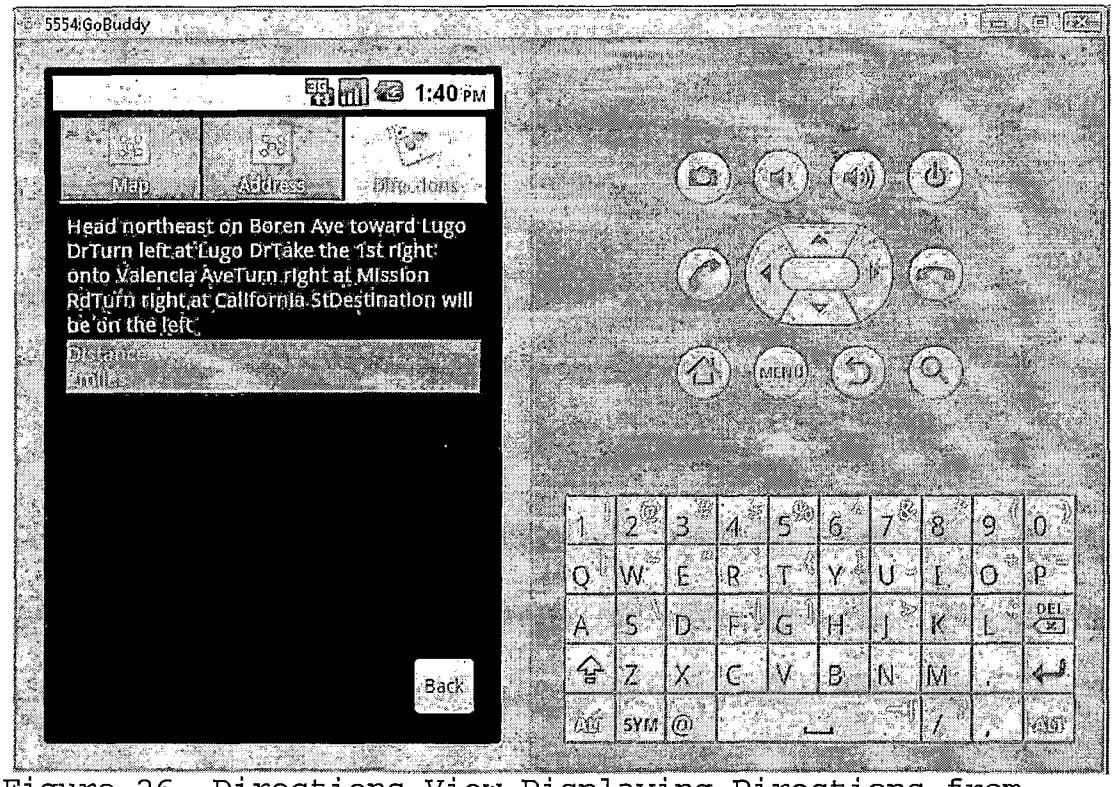


Figure 26. Directions View Displaying Directions from Current Location to the Location Displayed in Map View

4.6.2 Restaurants Icon

When the user clicks the Restaurants icon a http request call is made to Yahoo local search API and the query performs a search for restaurants that are in close proximity to the user's current latitude and longitude and returns a results set that are basically sorted by distance. The http response is an XML document, which is then parsed using a DOM (Document Object Model) parser. The results are then displayed in a tabbed view. The latitude and the longitude obtained from the query results

are used to form the GeoPoint. A marker is created and placed on the user's current latitude and longitude and on the latitude and longitude of the first item in the result set returned by the Yahoo local search API.
Simultaneously, the name, address, and phone number of the corresponding result item is placed on the address view and the directions from the user's current location to the place is also displayed on the directions tab. The user gets all the information about a place of interest all at the same time. The user just has to click on the various tabs to get the information he is looking for. The number of results returned by the search is limited to 10.

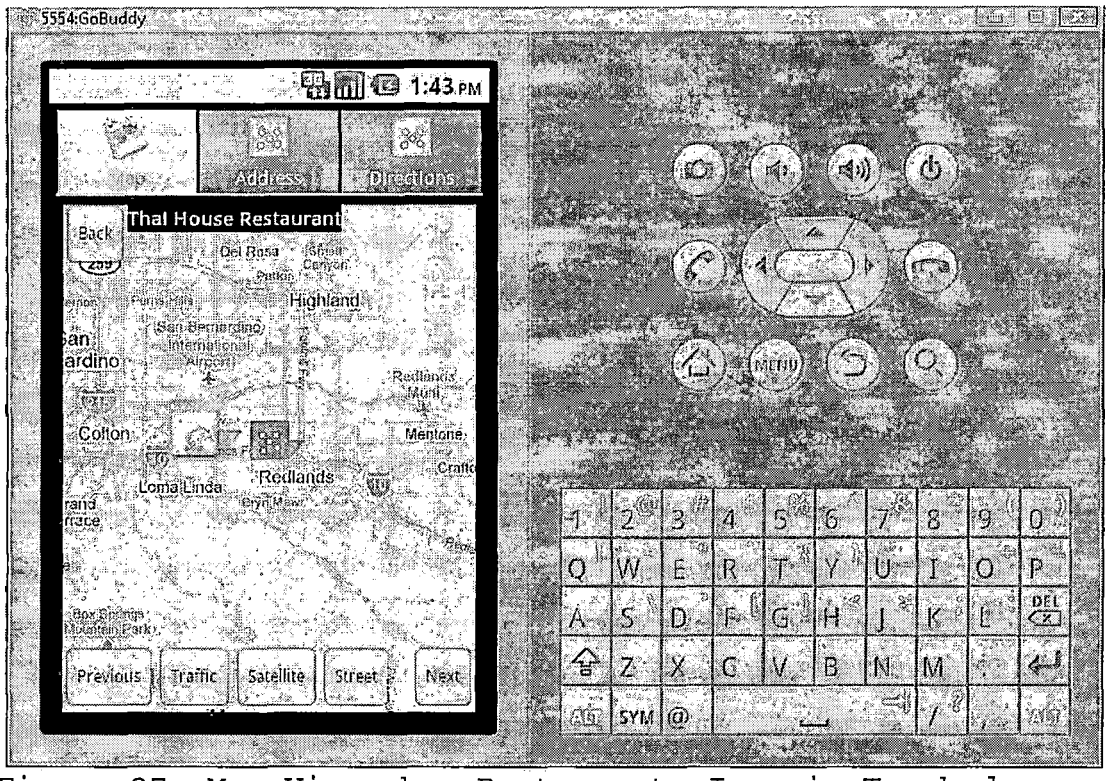


Figure 27. Map View when Restaurants Icon is Touched

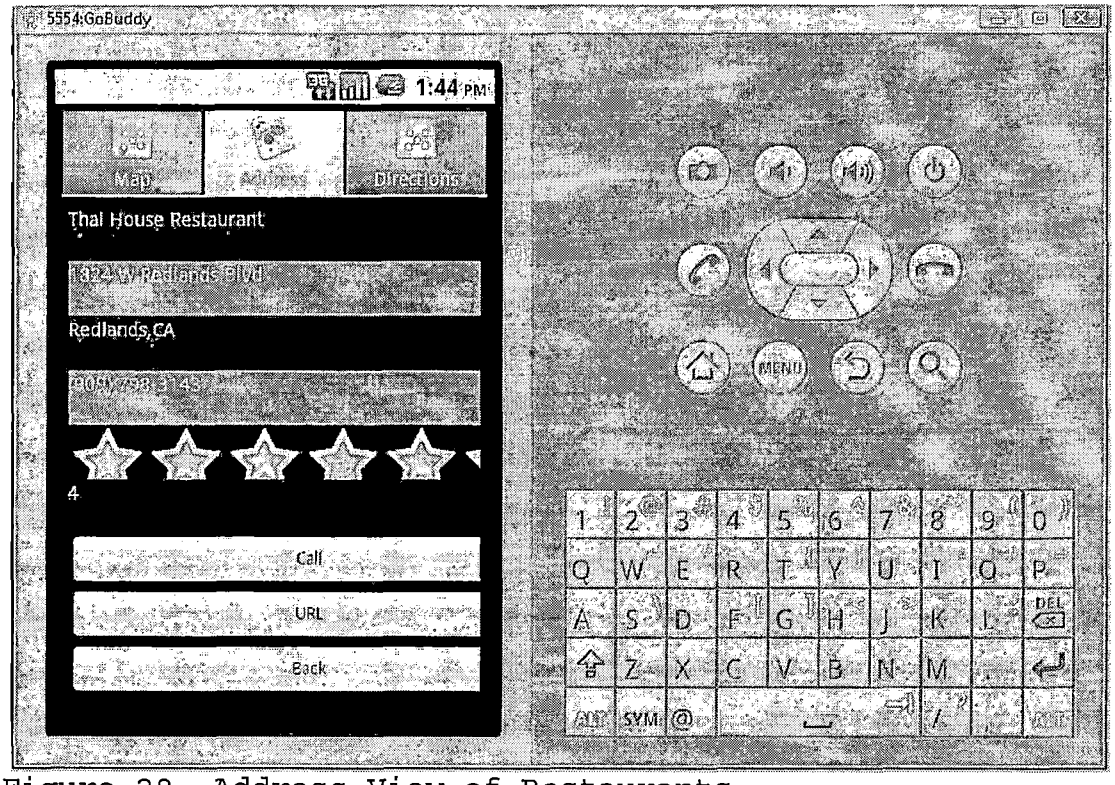


Figure 28. Address View of Restaurants

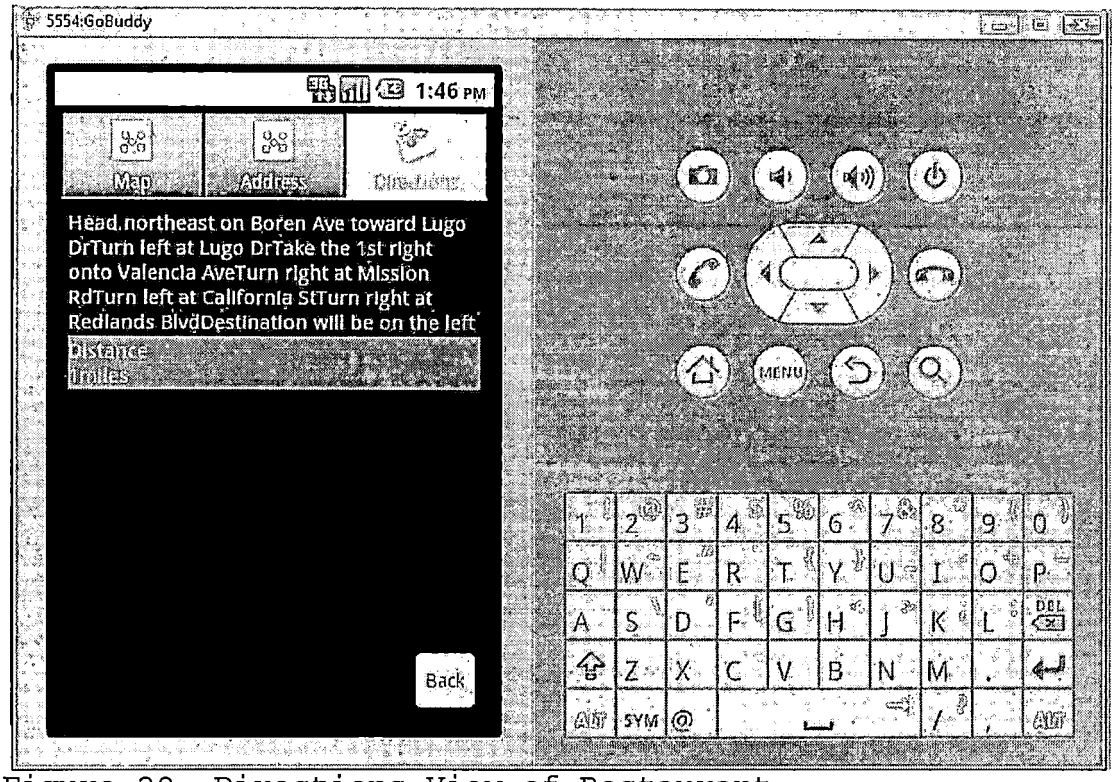


Figure 29. Directions View of Restaurant

4.6.3 Fast Foods

When the user clicks the Fast Foods icon a http request call is made to Yahoo local search API and the query performs a search for all the fast foods which includes both the chain and non chain types that are in close proximity to the user's current latitude and longitude and returns a results set that are basically sorted by distance. The http response is an XML document, which is then parsed using a DOM (Document Object Model) parser. The results are then displayed in a tabbed view. The latitude and the longitude obtained from the query

results are used to form the GeoPoint. A marker is created and placed on the user's current latitude and longitude and on the latitude and longitude of the first item in the result set returned by the Yahoo local search API. Simultaneously, the name, address, and phone number of the corresponding result item is placed on the address view and the directions from the user's current location to the place is also displayed on the directions tab. The user gets all the information about a place of interest all at the same time. The user just has to click on the various tabs to get the information he is looking for. The number of results returned by the search is limited to 10.

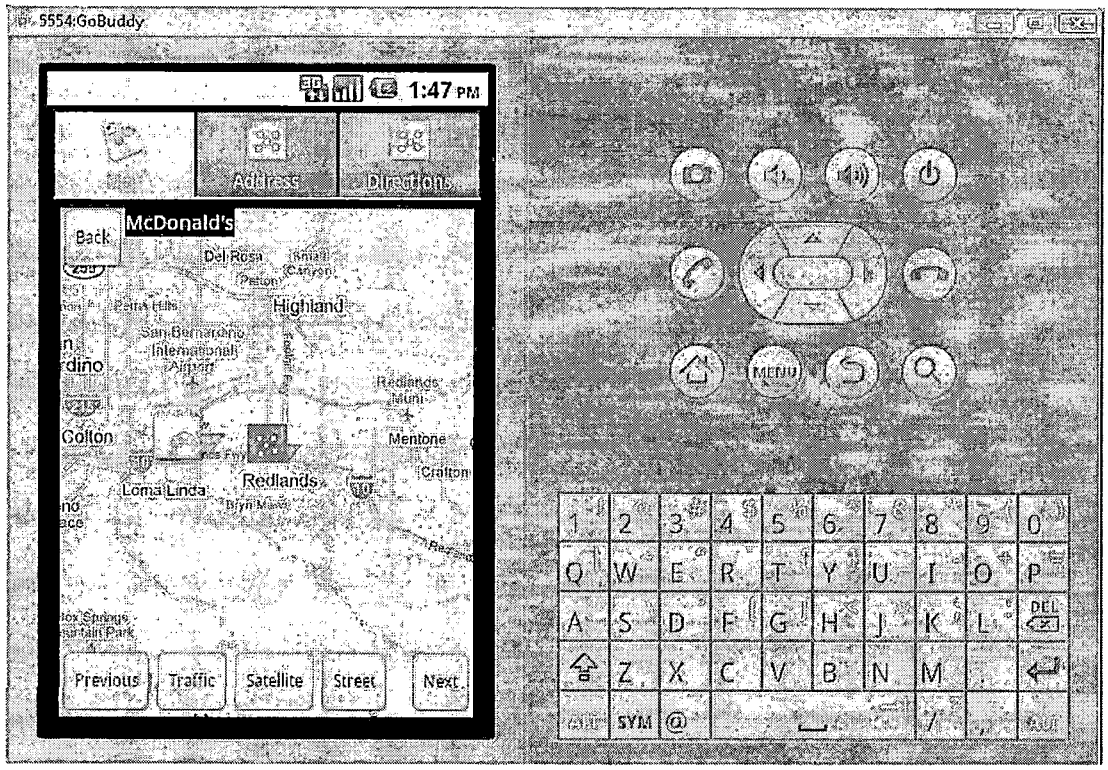


Figure 30. Map View for Fast Foods

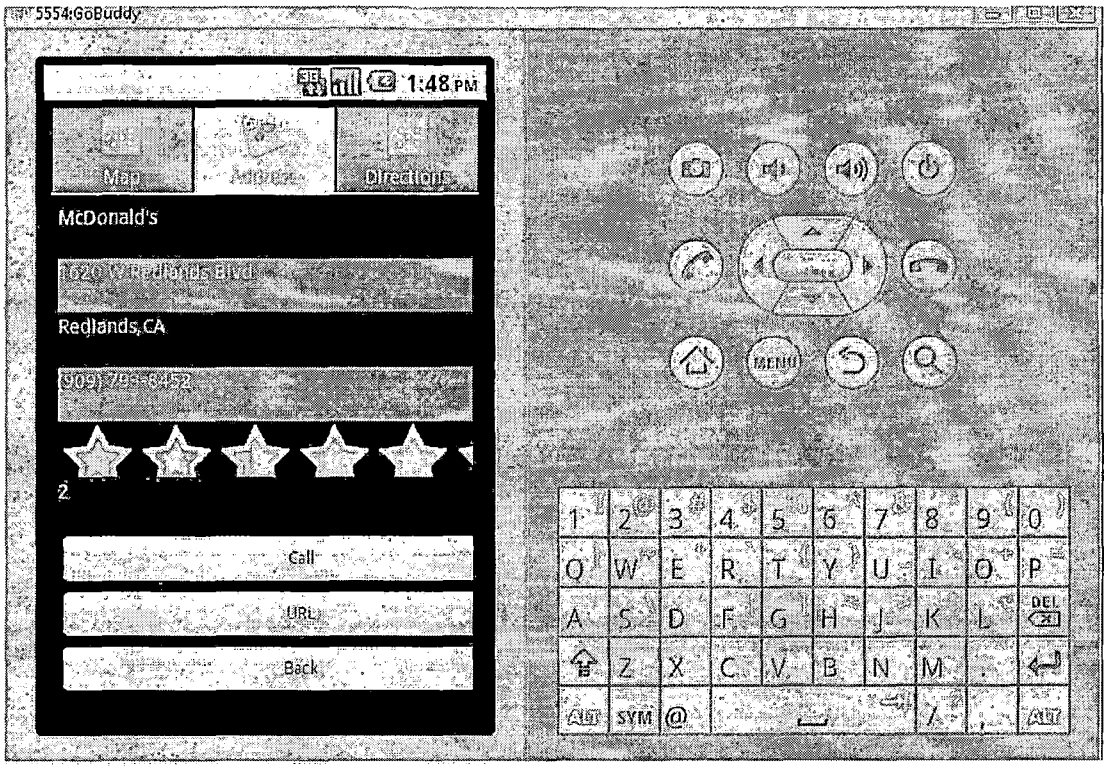


Figure 31. Address View for Fast Foods

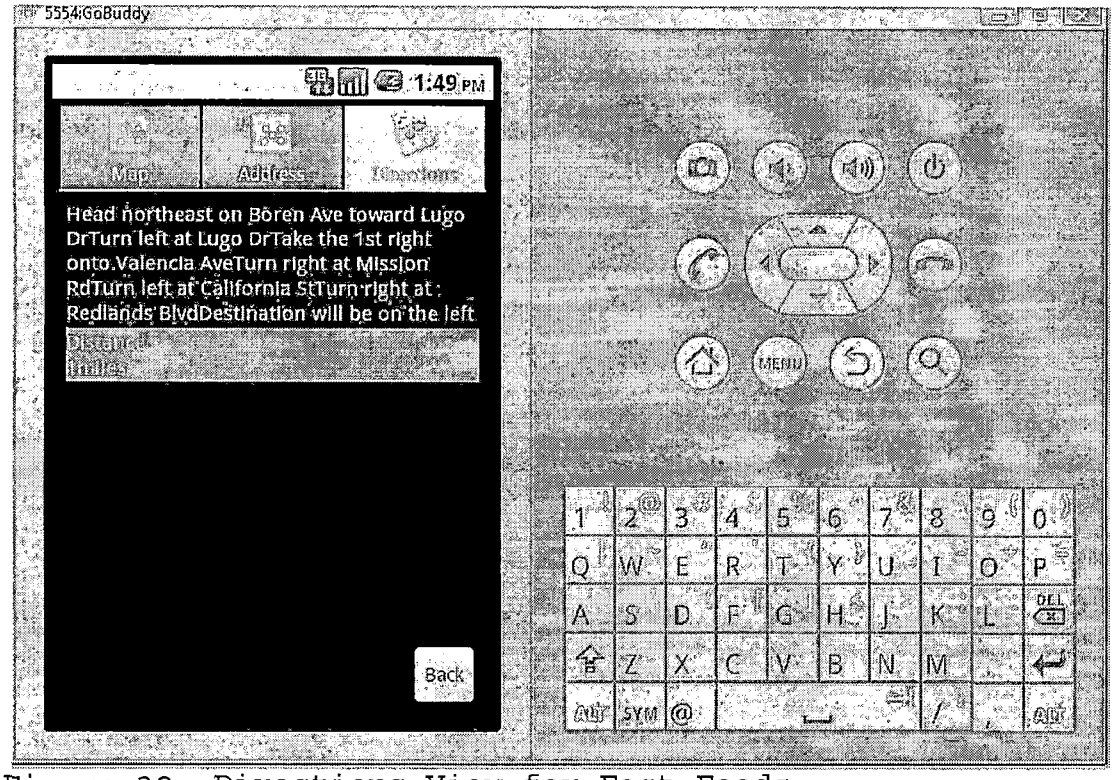


Figure 32. Directions View for Fast Foods

4.6.4 Gas Stations

When the user clicks the Gas Station icon a http request call is made to Yahoo local search API, and the query performs a search for all the gas stations around the area. The http response is an XML (Extensible Markup Lanuage) document, which is then parsed using a DOM (Document Object Model) parser. The results are then displayed in a tabbed view. The latitude and the longitude obtained from the query results are used to form the GeoPoint. A marker is created and placed on the user's current latitude and longitude and on the latitude and

longitude of the first item in the result set returned by the Yahoo local search API (Application Programming Interface). Simultaneously, the name, address, and phone number of the corresponding result item is placed on the address view and the directions from the user's current location to the place is also displayed on the directions tab. The user gets all the information about a place of interest all at the same time. The user just has to click on the various tabs to get the information he is looking for. The number of results returned by the search is limited to 10.

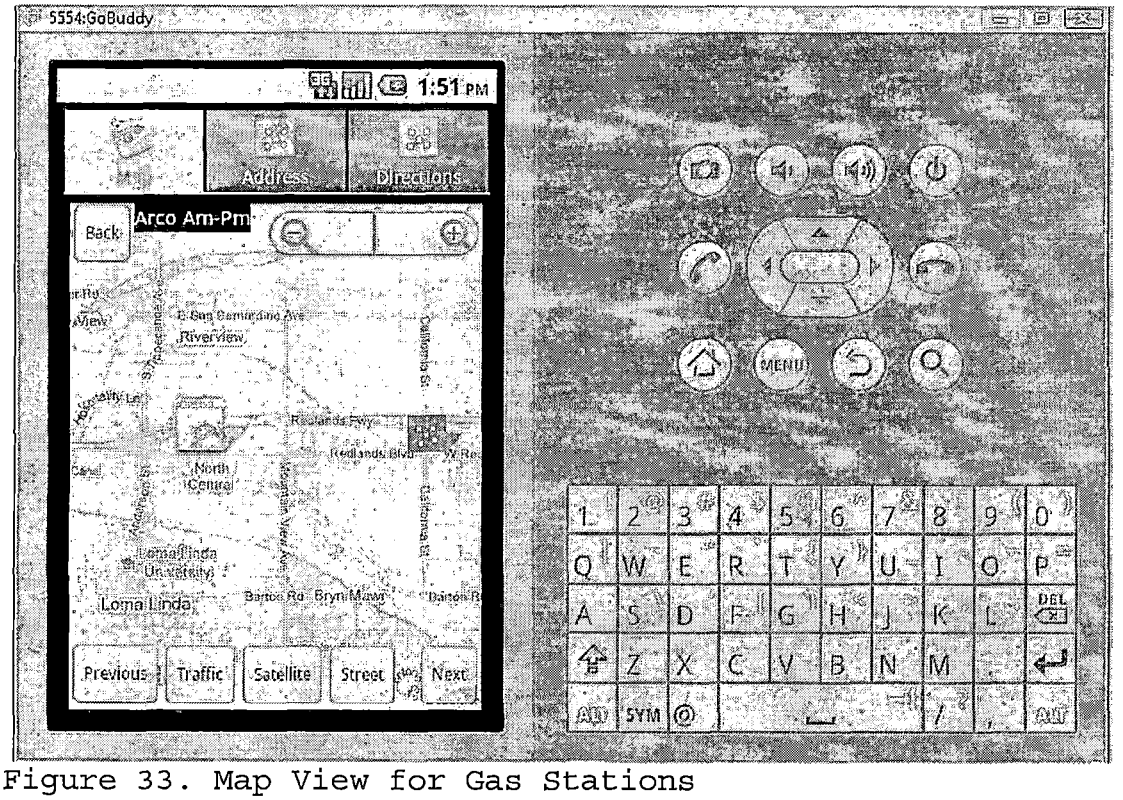


Figure 33. Map View for Gas Stations

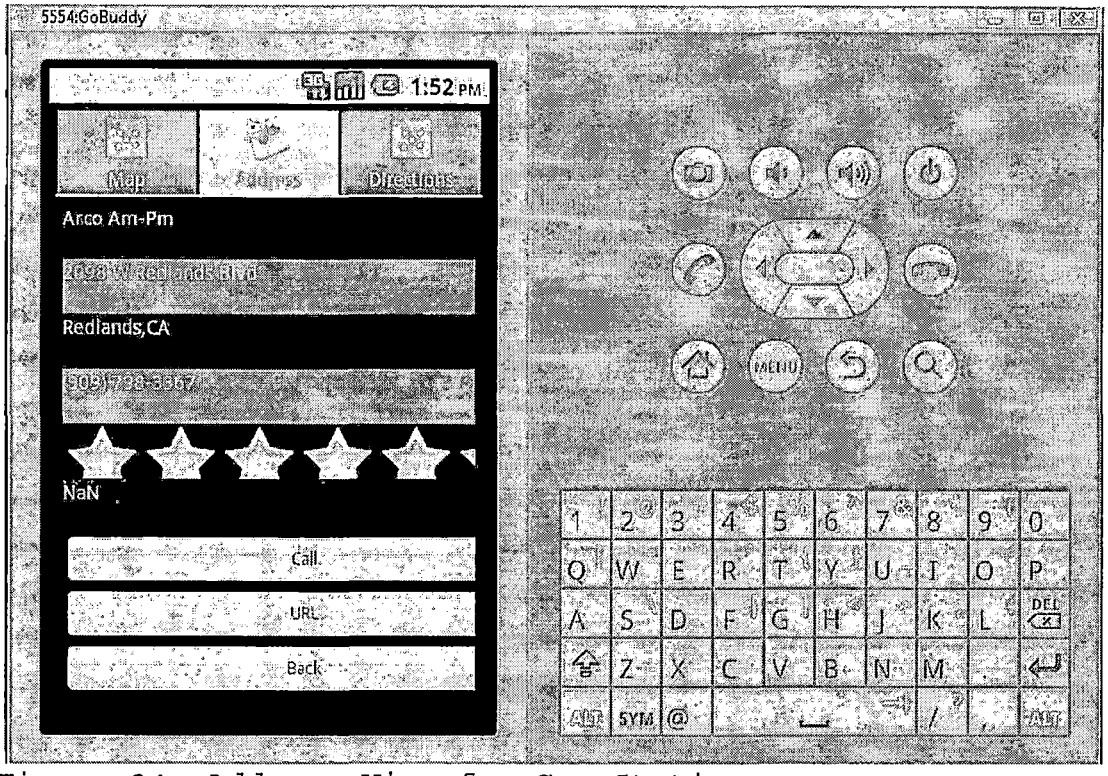


Figure 34. Address View for Gas Stations

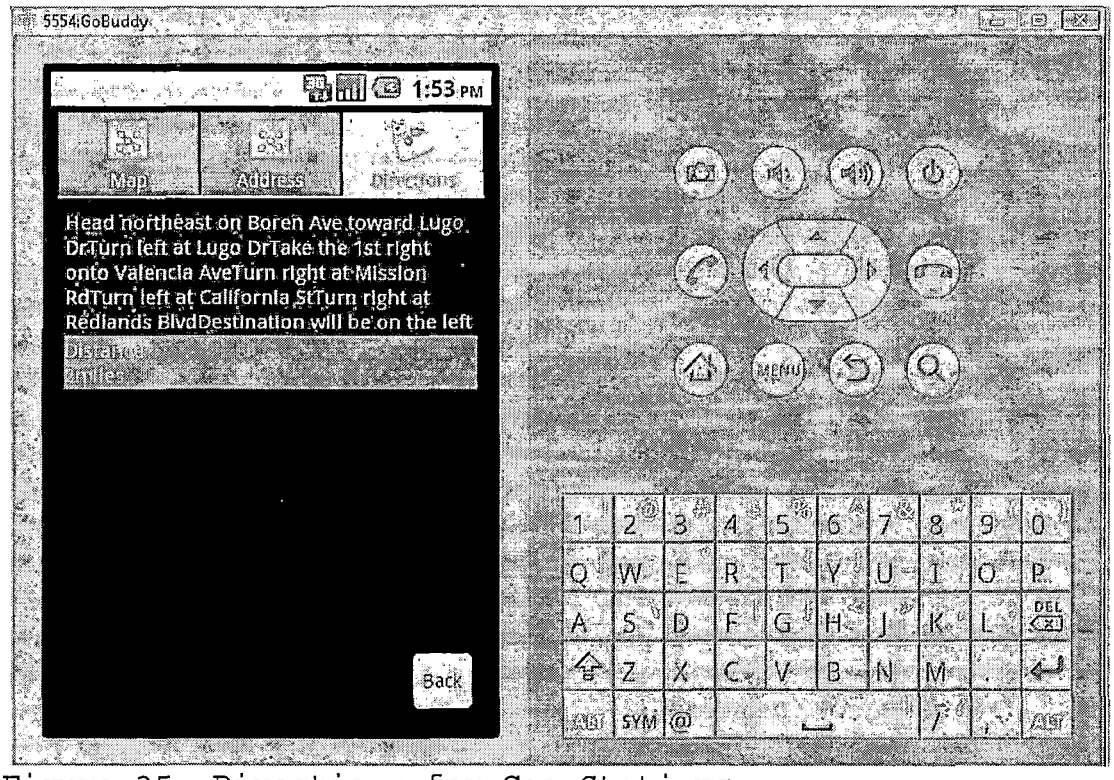


Figure 35. Directions for Gas Stations

4.6.5 Hotels

When the user clicks the Hotels icon a http request call is made to Yahoo local search API (Application Programming Interface) and the query performs a search for all the hotels in the nearby area. The http response is an XML (Extensible Markup Language) document, which is then parsed using a DOM (Document Object Model) parser. The results are then displayed in a tabbed view. The latitude and the longitude obtained from the query results are used to form the GeoPoint. A marker is created and placed on the user's current latitude and longitude and on the

latitude and longitude of the first item in the result set returned by the Yahoo local search API (Application Programming Interface. Simultaneously, the name, address and phone number of the corresponding result item is placed on the address view and the directions from the user's current location to the place is also displayed on the directions tab. The user gets all the information about a place of interest all at the same time. The user just has to click on the various tabs to get the information he is looking for. The number of results returned by the search is limited to 10. Having the link to the website a s part of the information repository makes it easier for the end user to check the rates and availability also.

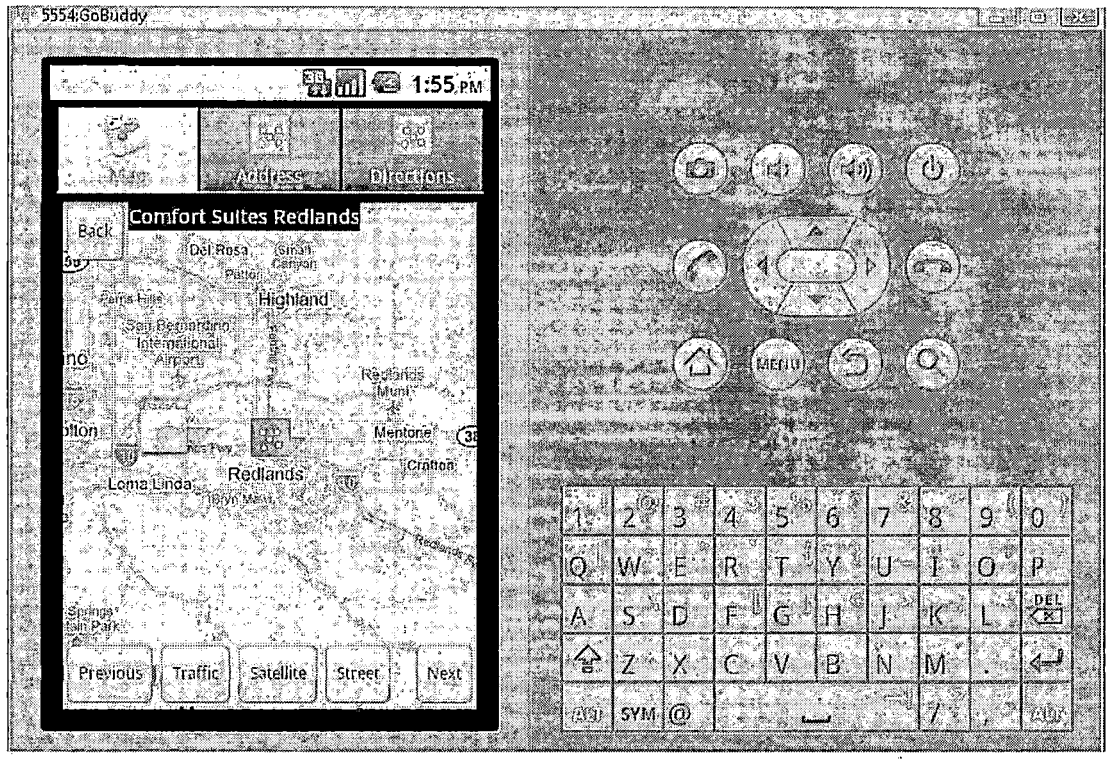


Figure 36. Map View for Hotels

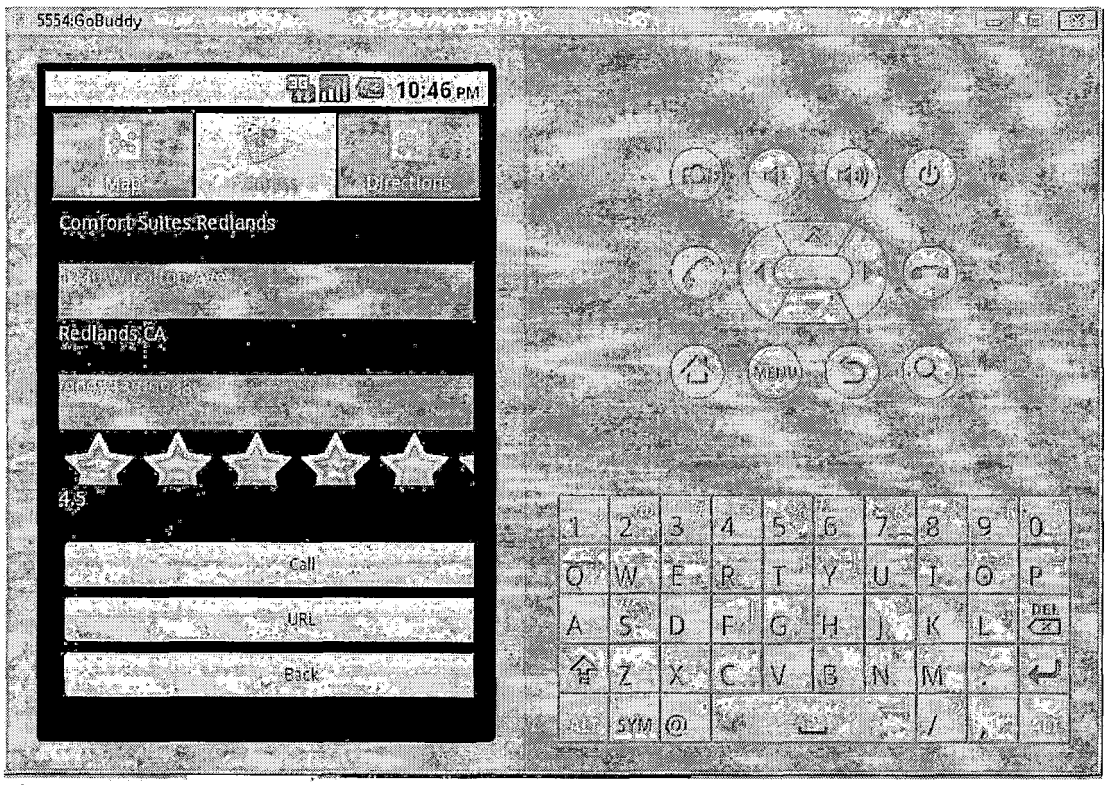


Figure 37. Address View for Hotels

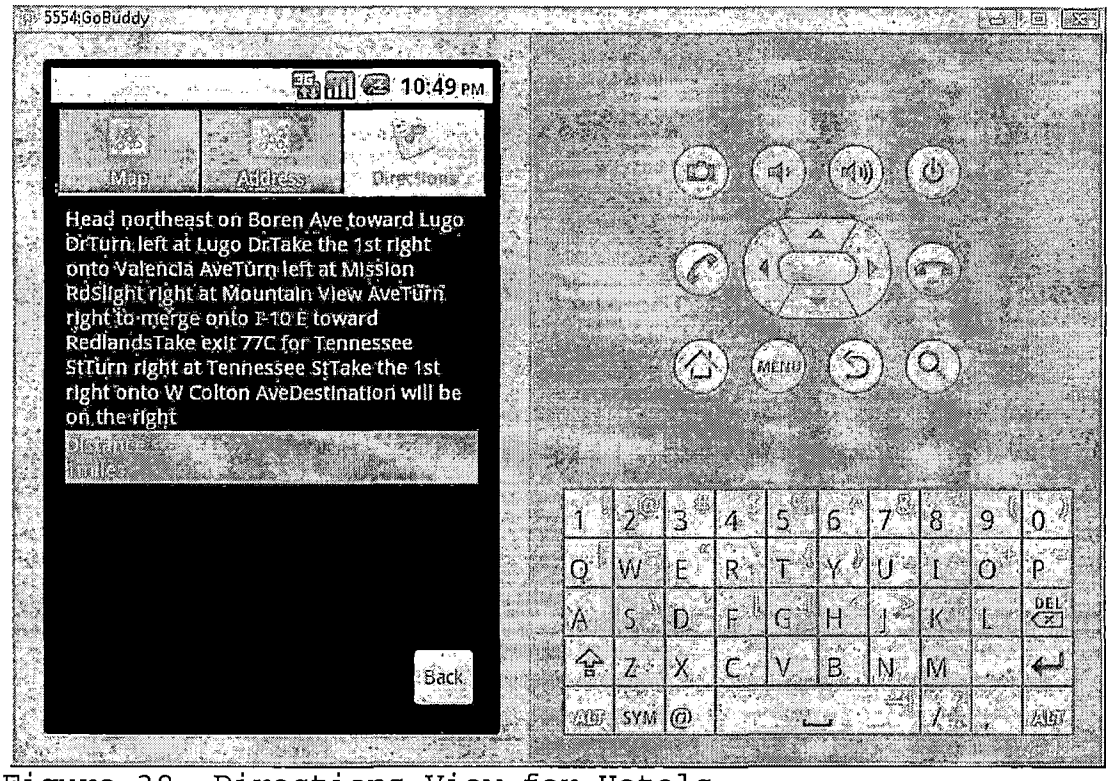


Figure 38. Directions View for Hotels

4.6.6 Movie Theaters

When the user clicks the Movie Theaters icon a http request call is made to Yahoo local search API and the query performs a search for the closest movie theaters around the area. The http response is an XML document, which is then parsed using a DOM (Document Object Model) parser. The results are then displayed in a tabbed view. The latitude and the longitude obtained from the query results are used to form the GeoPoint. A marker is created and placed on the user's current latitude and longitude and on the latitude and longitude of the first item in the

result set returned by the Yahoo local search API. Simultaneously, the name, address, and phone number of the corresponding result item is placed on the address view and the directions from the user's current location to the place is also displayed on the directions tab. The user gets all the information about a place of interest all at the same time. The user just has to click on the various tabs to get the information he is looking for. The number of results returned by the search is limited to 10.

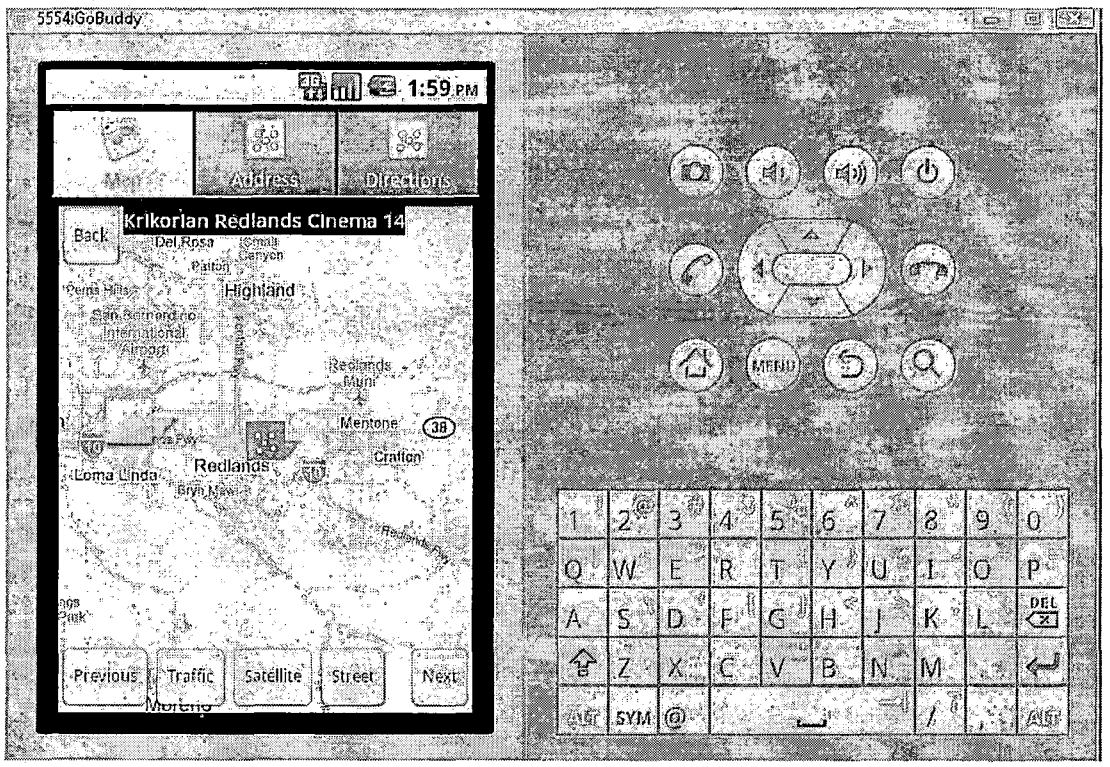


Figure 39. Movie Theaters Map View

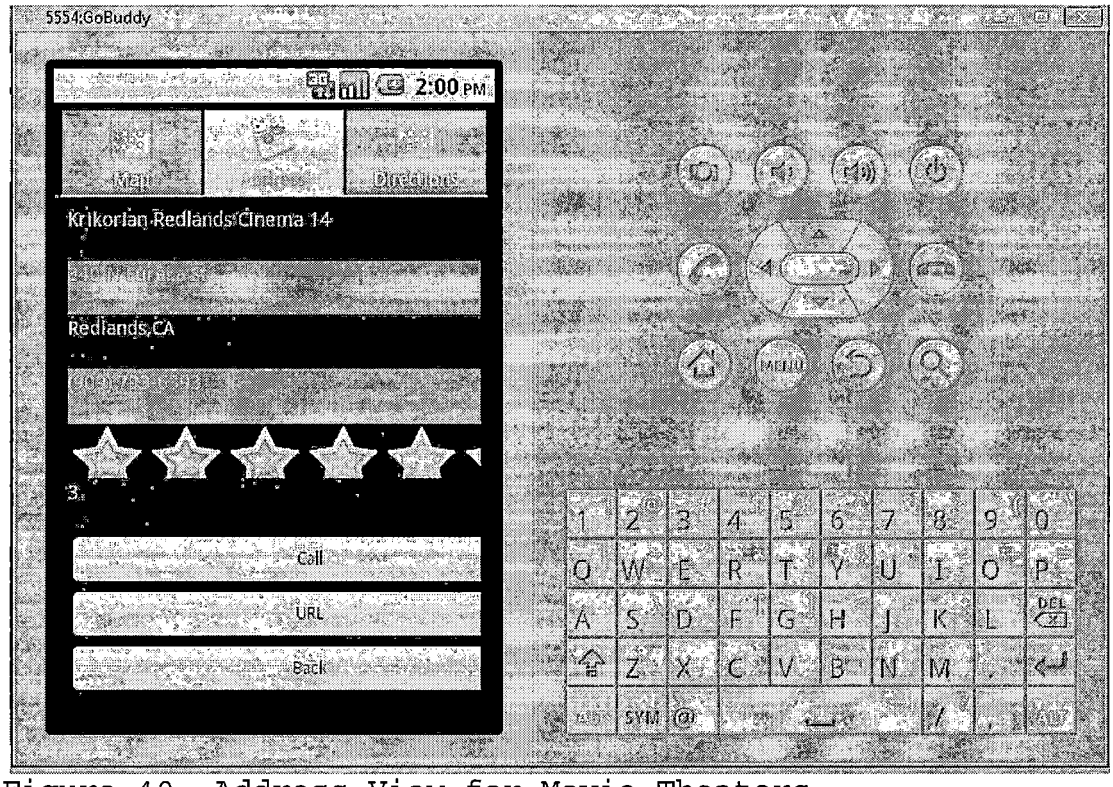


Figure 40. Address View for Movie Theaters

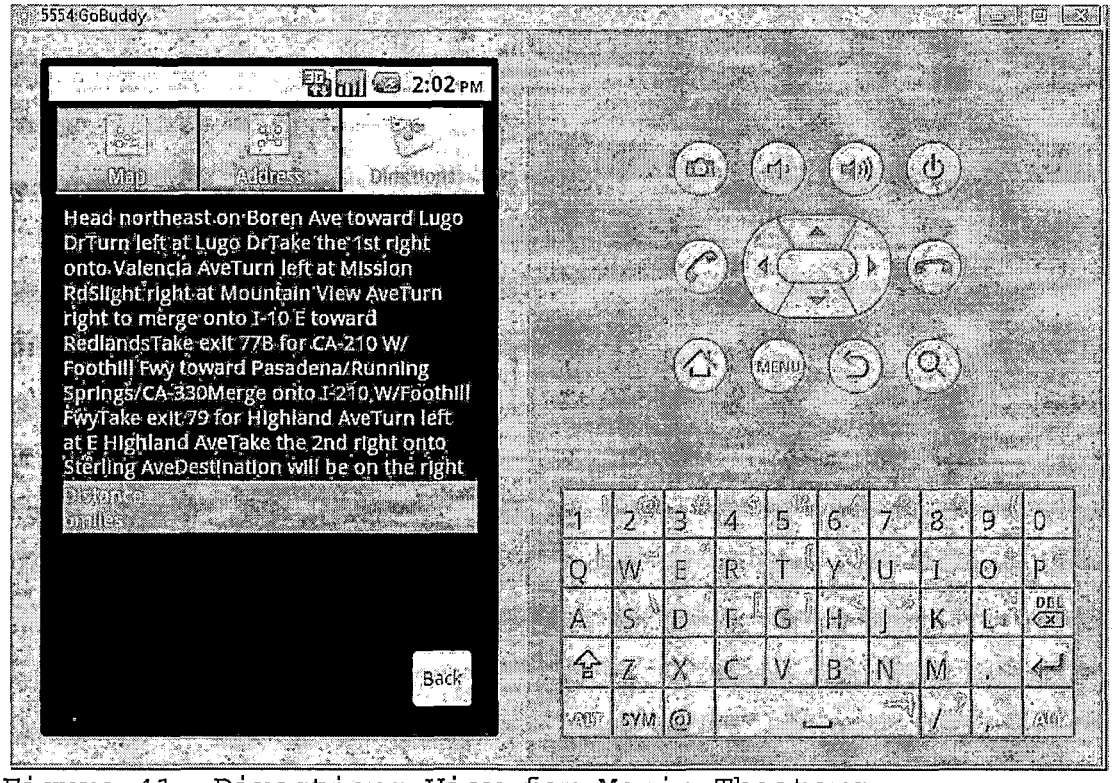


Figure 41. Directions View for Movie Theaters

4.6.7 Attractions Icon

When the user clicks the Attractions icon a http request call is made to Yahoo local search API and the query performs a search for all the attractions in the location. The http response is an XML document, which is then parsed using a DOM (Document Object Model) parser. The results are then displayed in a tabbed view. The latitude and the longitude obtained from the query results are used to form the GeoPoint. A marker is created and placed on the user's current latitude and longitude and on the latitude and longitude of the first item in the result

set returned by the Yahoo local search API. Simultaneously, the name, address and phone number of the corresponding result item is placed on the address view and the directions from the user's current location to the place is also displayed on the directions tab. The user gets all the information about a place of interest all at the same time. The user just has to click on the various tabs to get the information he is looking for.

4.6.8 Search Icon

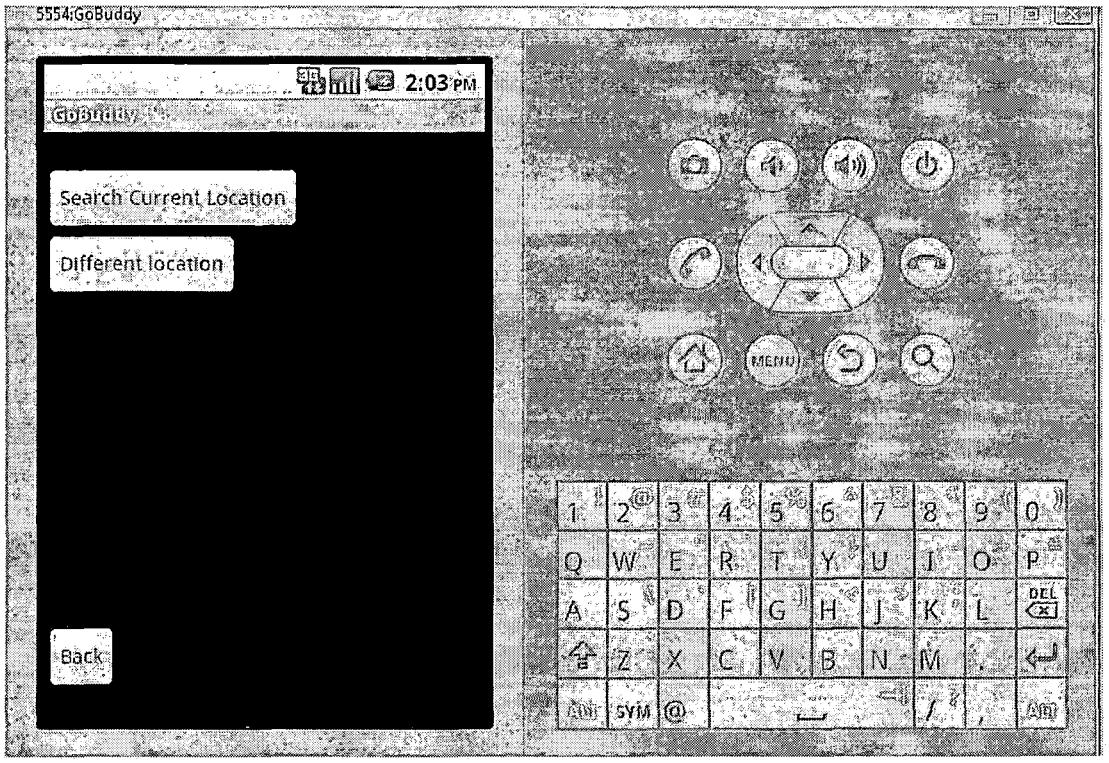


Figure 42. When the Search Icon on the Main Screen is Clicked

When the user touches the search icon in the main menu, the screen is navigated to another view. Here the user has to choose if he wants to perform a search in the current location or in a different location.

4.6.8.1 Search Current Location. The screen when the user clicks the search current location button. Then, the user enters the search criteria and touches the search button. This performs a http request and returns the response in a tabbed view like the previous one.

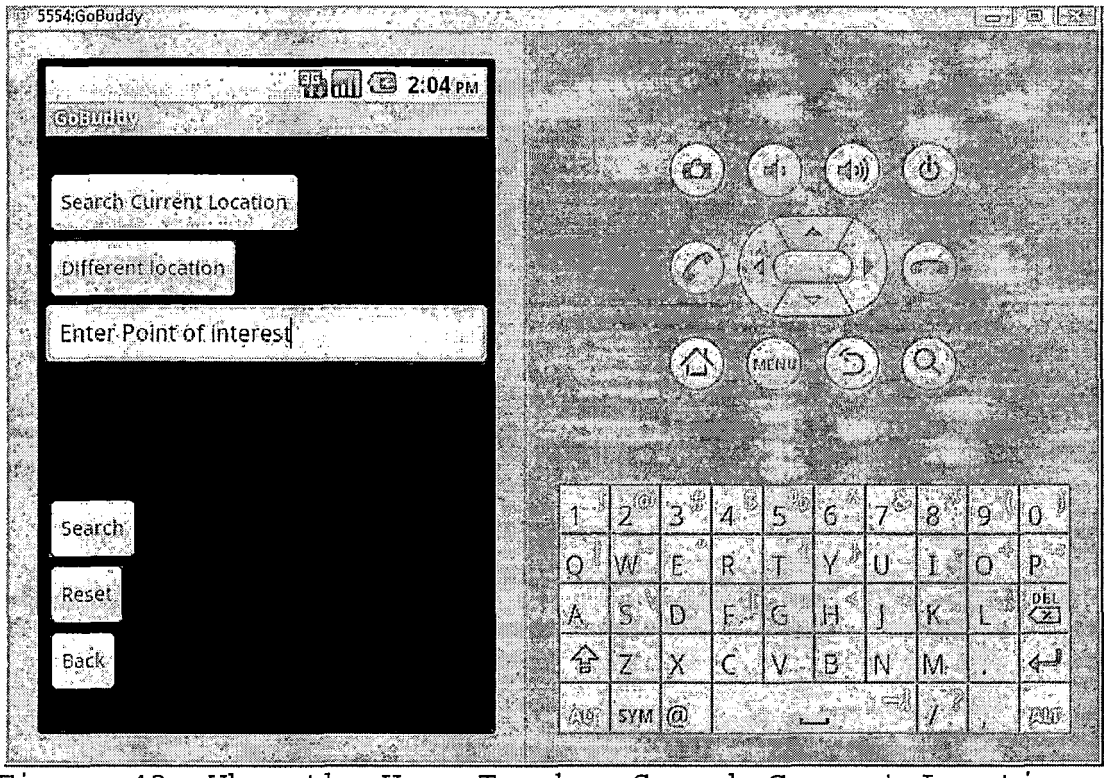


Figure 43. When the User Touches Search Current Location

4.6.8.2 Search Different Location. The screen when the user clicks the search different location button. The user has to enter the search text, name of the city and select the state from the provided list view. When the user touches the search button, the query results are retrieved and displayed in a tabbed view.

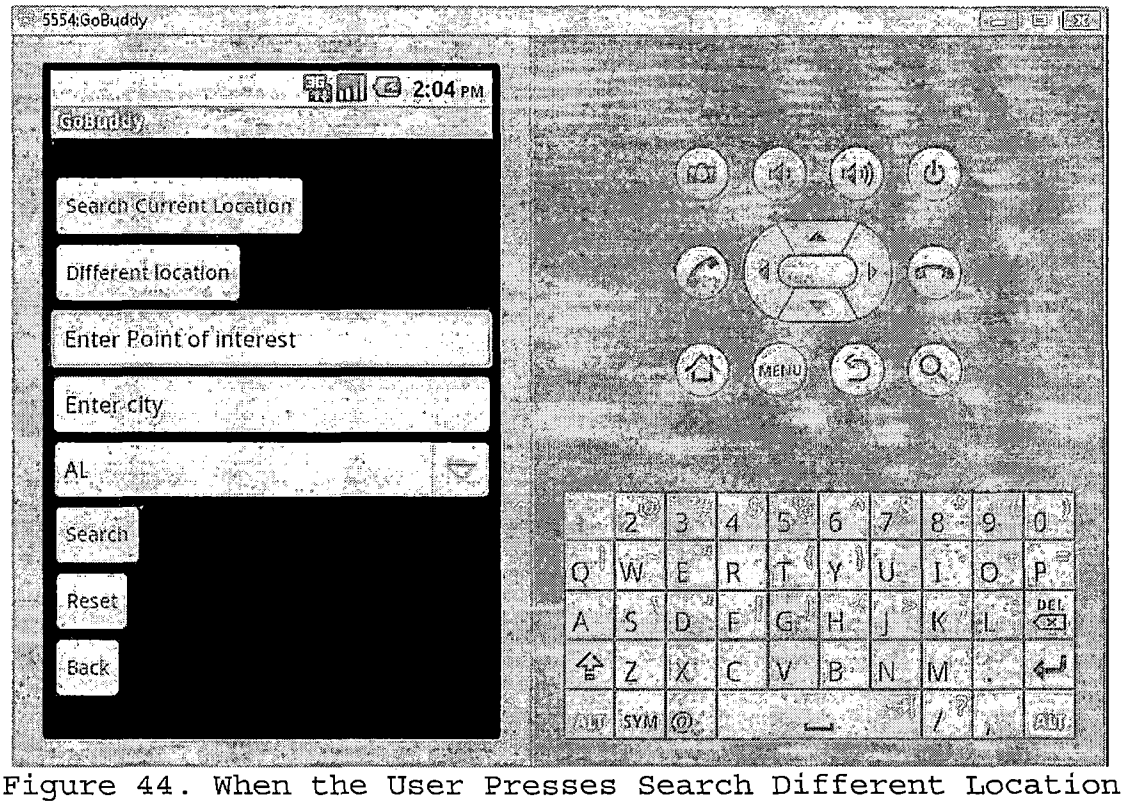


Figure 44. When the User Presses Search Different Location Button

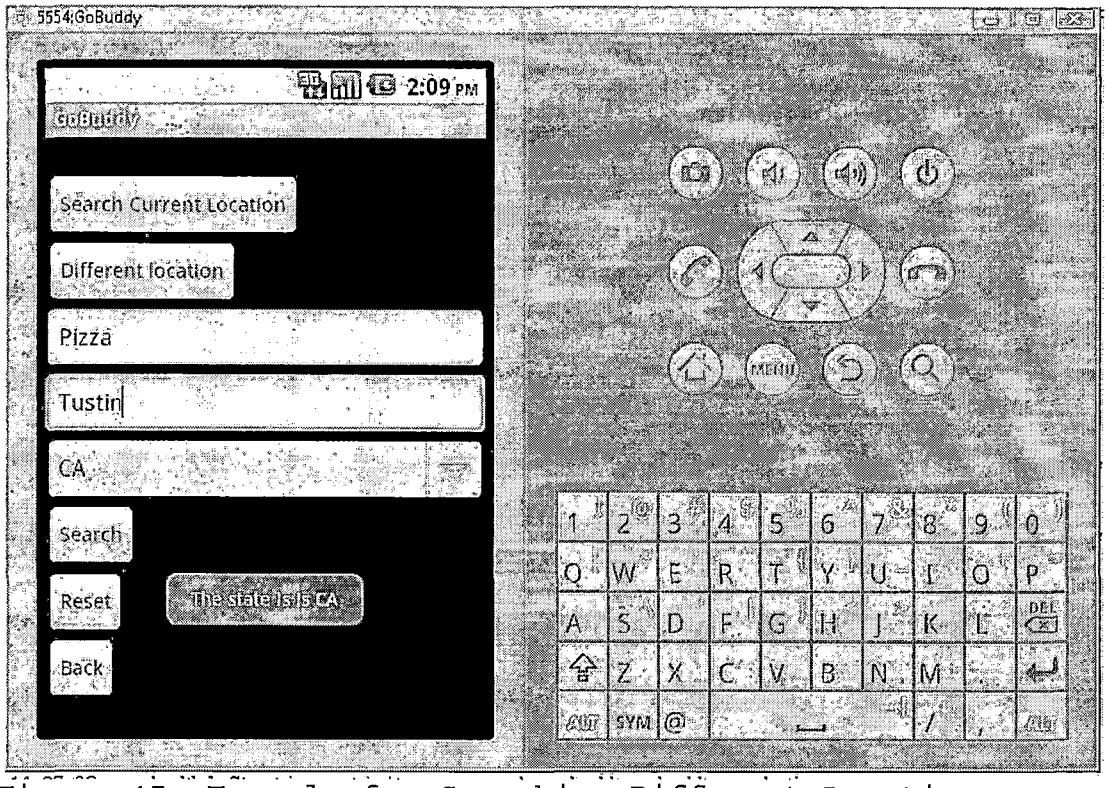


Figure 45. Example for Searching Different Location

4.7 Map Views

4.7.1 Street View

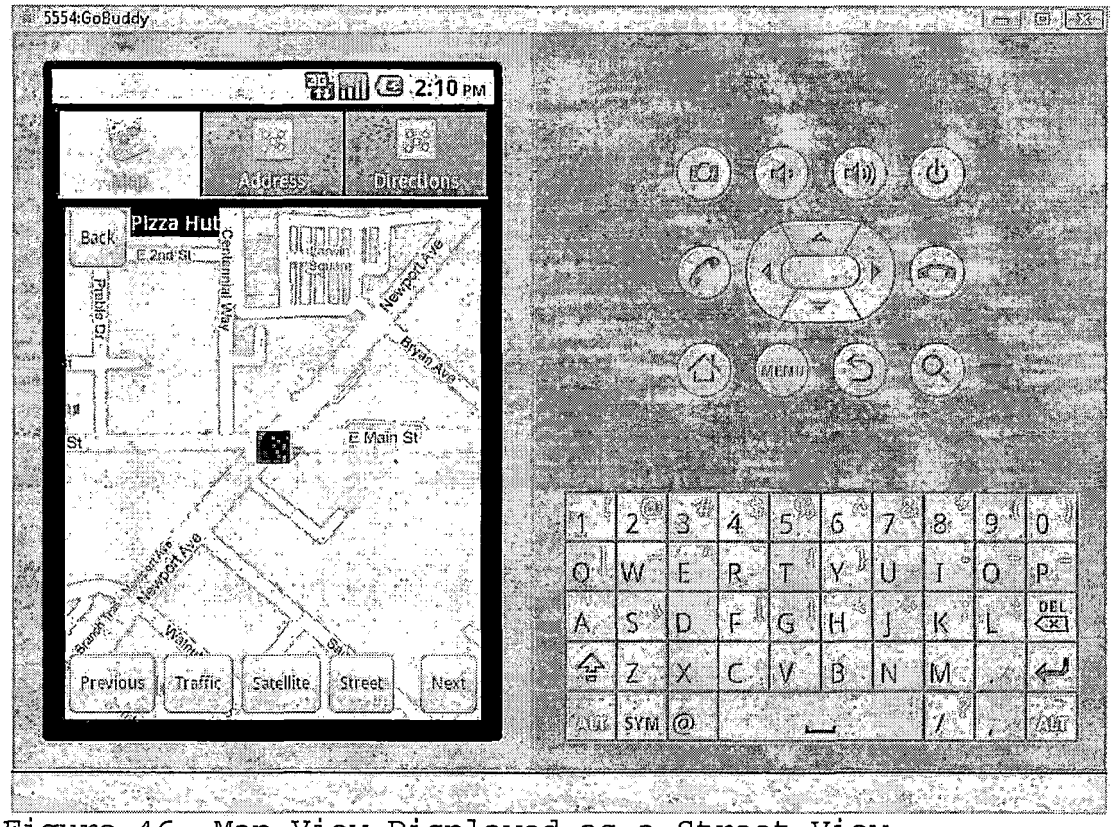


Figure 46. Map View Displayed as a Street View

4.7.2 Satellite Button

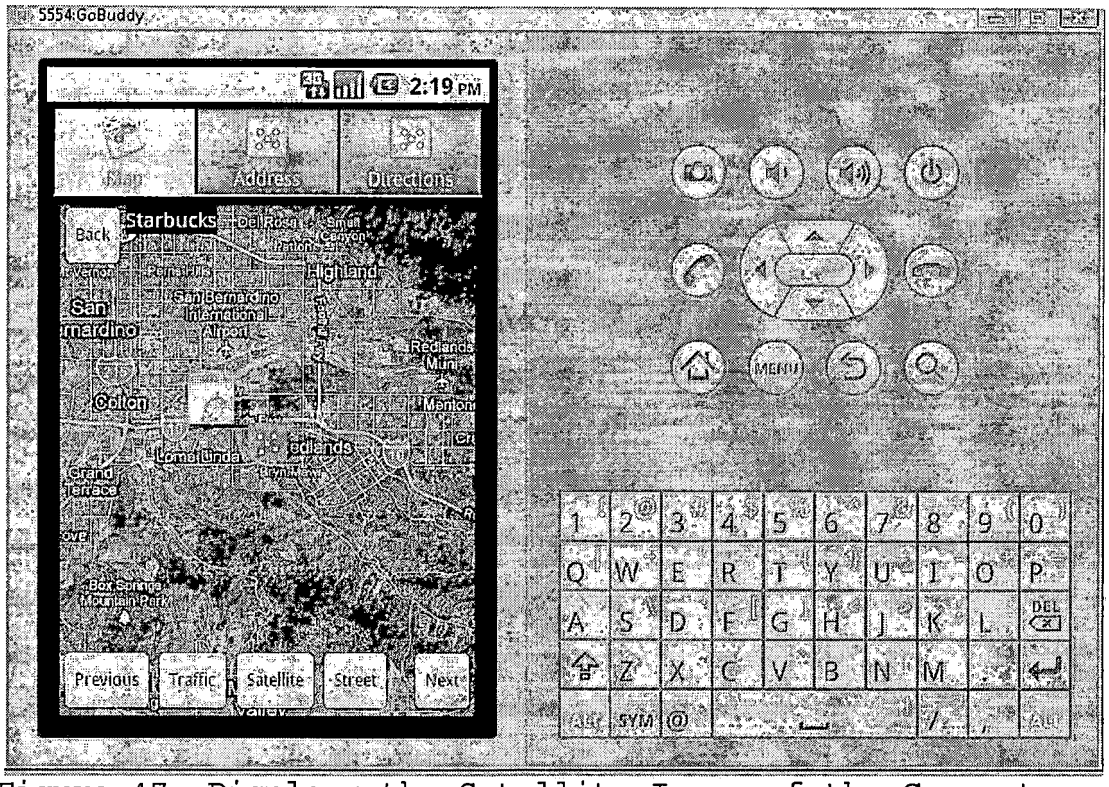


Figure 47. Displays the Satellite Image of the Current Location on the Map

4.7.3 Traffic Button

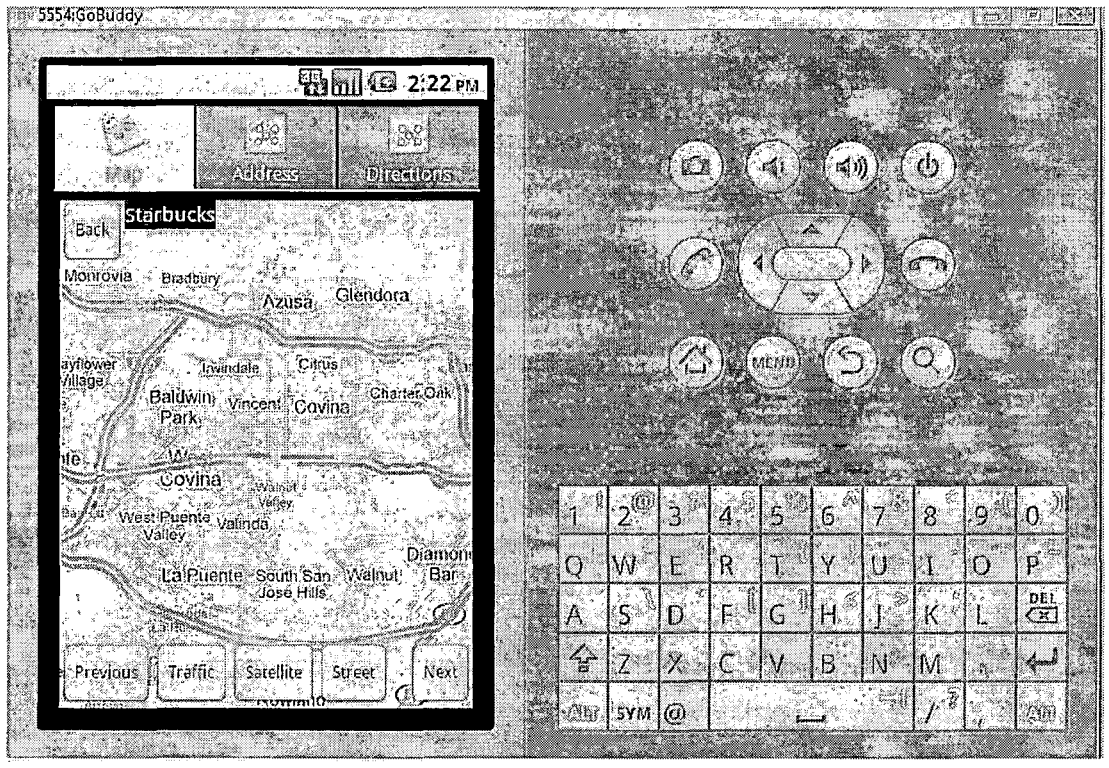


Figure 48. Displays the Traffic Conditions with Respect to the Current Location

4.8 Call Button

This initiates a phone call to a particular place of interest.

4.9 Uniform Resource Locator (URL) Button

Navigates the screen to a web view where the website of the place is displayed. Since this view is initiated outside the tab activity, the back button on the phone is

used to go back to the previous screen instead of having a separate back button.

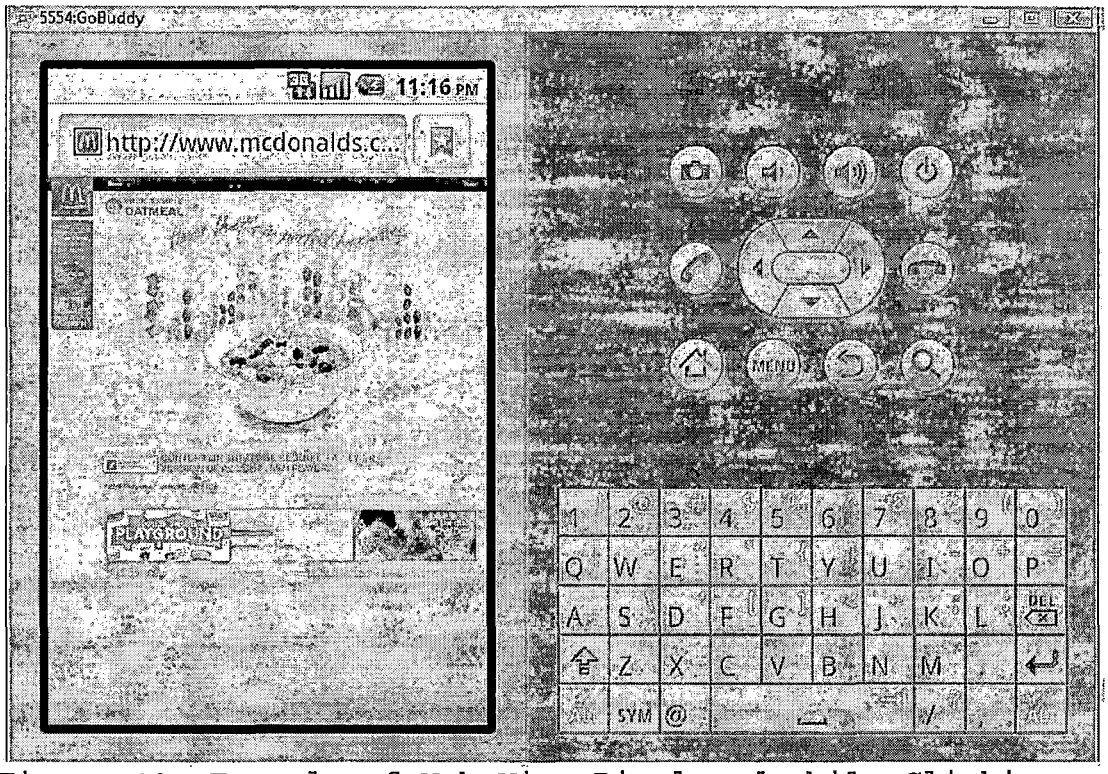


Figure 49. Example of Web View Displayed while Clicking the Uniform Resource Locator (URL) Button

The user can navigate between the screen as easy as with a single touch or tap. The pre-defined image icons in the main screen makes it very easy for the user to find the closest places. Also, the results are displayed as list of map view rather than a text based list view, which makes the application more appealing to the user. The buttons provided in each of the views provides an

attractive set of options which makes the application rich and wholesome. On the whole, this application makes it easy for an user, whether he is travelling or planning for a visit to a place in the United States. The main advantage of this mobile application over browsing the internet for information is that, the user can interact with the application even when he is on the move or outside and need not be dependent on internet connection or a computer.

CHAPTER FIVE

SOFTWARE TESTING

Android offers a powerful and easy-to-use testing frame work that is well integrated with the Eclipse IDE. It provides a consistent way of testing framework that can be used to test each unit. Android test suites are based on JUnit testing that provides test cases to test every aspect of the application. The test methods are bundled up into test classes and each testing is done to examine each individual module of the application. The test files are organized into projects that include the source code, android manifest file and other resources. The two base classes that are needed to perform testing Android applications are AndroidTestCase, Assert, and InstrumentationTestRunner. The AndroidTestCase class has the setup, teardown and helper methods for checking entire components life cycle, Assert class has methods used for displaying the test results and the InstrumentationTestRunner is used for running the test case.

5.1 JUnit Testing of GoBuddy

GoBuddy was tested using the Android JUnit test tools available within the Eclipse IDE. A test project was

created in the same directory as that of the main application. The test project consisted of the android manifest file and the source code containing test cases. Several test cases were generated and ran to check the various modules and their functionalities. Once the test application was run, the JUnit window was displayed and it displays the test cases that were successful and those of which failed. For JUnit test cases and test results Refer Appendix B.

CHAPTER SIX

MAINTENANCE MANUAL

This explains the tools and system requirements to develop or enhance the android application on Eclipse IDE. It also has details on how to deploy the application on a mobile device.

6.1 System and Software Requirements for Developing Android Applications

System Requirements

6.1.1 Operating System Used

Microsoft Windows Vista

6.1.2 Development Environment

Eclipse Helios

Eclipse JDT (Java Development Tools)

JDK 1.5

Android development Tools Plug-in

6.1.3 Hardware Requirements

The Android SDK requires disk space to store the components needed for developing the application. In addition to the Eclipse IDE, JDk the disk space required are as follows

SDK Tools - 35 MB

SDK Platform Tools - 6 MB

Android platform -150 MB

6.2 Downloading the Required Software and Tools

6.2.1 Java (JDK5 or JDK6)

1. Open the link
   http://java.sun.com/javase/downloads/5u22/jdk
   (Version JDK1.5).
2. Select the platform as Windows.
3. Download and install JDK

6.2.2 Eclipse Integrated Development Environment

1. Open the link
   http://www.eclipse.org/downloads/packages/
   eclipse-ide-java-developers/heliossr1
2. Download the zip file to a directory.
3. Extract and install Eclipse helios

6.2.3 Android Software Development Kit

- The android SDK starter package must be downloaded from the following link
  http://developer.android.com/sdk/index.html
  Select Windows and download the file and install the SDK starter packager. It is optional to download the Windows installer which might help with the installation.

- Install and configure the Android Development Tools (ADT) plug-in for Eclipse and configure it by following the instructions in the link http://developer.android.com/sdk/eclipse-adt.html#installing
- Download essential SDK components for developing the application
    - Setting up the Android SDK and AVD Manager that comes with the Android SDK starter package. This is needed to further download and install the components need for a basic functioning of the Android SDK.
    - Install the Android SDK and AVD Manager
    - Launch the Android SDK and AVD manager by selecting Window→Android SDK and AVD Manager from within Eclipse.

There are two repositories - one from the Android SDK and the other one from third parties. At least one Android platform should be downloaded so that the application can be compiled and the AVD (Android Virtual Device) can be set up to run the application on the android emulator.

Since Travel Buddy is a location based application that is dependent on Google, google API was selected and downloaded.

6.2.4 Creating the Android Virtual Device (AVD)

The Android Virtual Device is created by selecting the Android SDK and AVD Manager.

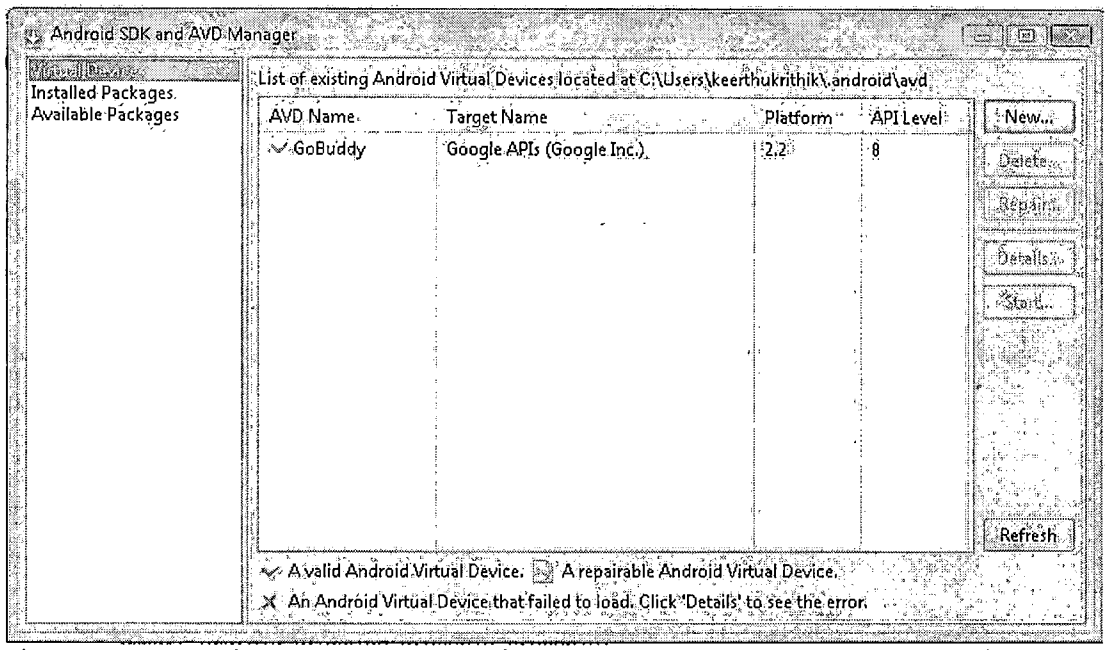


Figure 50. Diagram - Android Software Development Kit and Android Virtual Device Manager

Now the total development environment is ready for developing, compiling, and running Android applications.

6.3 Configure Google to use the Google Maps Application Programming Interface (API)

Even though the Google API for Android was installed using the Android SDK and AVD Manager, it still cannot make use of the API features. It is required to get a Google API key for using it with the application. The

Google API is key is provided by Google Inc. and has two versions- debug key and publishing key.

When developing the application, the debug key is sufficient to develop and test the applications. Once it is ready for deployment, an official publishing key has to be acquired from Google.

For development purposes, the debug key was acquired in the following manner.

http://code.google.com/android/add-ons/

google-apis/mapkey.html

Follow the instructions in the above link and by using the keytool command

c:\PragramFIles\java\jdk1.5...\bin\keytool is here

keytool -list - alias androiddebugkey -keystore C:\Users|keerthukrithik\.android\debog.keystore - storepass android -keypass android

GoogleMaps API Key generated

0doFvhRygehCfxzBmsv4RPXYiRKapltLKcivdZg

MD5 fingerprint

5D:32:DC:7B:A6:B7:EB:AF:2C:DF:2D:F8:F2:36:62:60

The Google Maps API key is then declared in the XML file as follows.

```
<com.google.android.maps.MapView
    android:layout_width="fill_parent"
```

```
android:layout_height="fill_parent"
android:apiKey="0doFvhRygehCfxzBmsv4RPXYiRKapltLKcivdZg
/>
```

6.4 Configure Google Directions Application Programming Interface (API)

It is not required to explicitly get Google directions API key. It can be used in the application with the Google Maps API Key generated earlier.

6.5 Configure Yahoo Local Search Application Programming Interface (API)

Getting the API key with Yahoo is much simpler and less tedious when compared to Google. Follow the link

https://developer.apps.yahoo.com/wsregapp/

Select Generic for the Authentication method and fill out the required information and the Yahoo API key is generated right away.

6.6 Deployment in the Real Device

Once the application has been developed and tested on the emulator, it can be deployed on to any Android smart phone. Download the USB driver from the website of the manufacturer of the phone. Find the driver based on the model number provided with the phone. Once the appropriate driver has been downloaded, connect the phone to the computer. Open Eclipse and select Run Configurations from

the Run tab. Here, select the target as the device and then click run once again. Now the .apk file will be installed onto the device and the application can be launched on the device by pressing the icon on the applications window on the real device.

CHAPTER SEVEN

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

GoBuddy is a location-based application that renders a very rich set of location-based information to the end-user. With the application in hand on a mobile device, the user is able to find and locate nearby coffee shops, restaurants, fast foods, gas stations, hotels, movie theaters and attractions with a single touch or tap on the device. These are considered as common places of interest. In addition the user is also given the feasibility of performing his own choice of interest in the same or different location within the United States. On the whole, GoBuddy serves as an easy to use, rich mobile application that acts as a place finder and travel guide to the end users.

7.2 Future Enhancements

Although the application has tried to encompass a whole bunch of information about a particular place of interest, the application can still be enhanced in many ways.

7.2.1 Increasing the Size of the Result Set

The web server used to do the local search is Yahoo local search API. Several attempts were made to perform ht local search though the Google Places API. But the regulations imposed by Google to provide the user with an authenticity is so tedious and time consuming, Yahoo local search API was used in place which has several limitations such as the maximum number of search results returned could only be 20. If Google's place API is used with the application, then the search will become more optimal and accurate.

7.2.2 Enhancing the User Interface

Even though the initial priority was to design the application in an attractive way, due to time constraints more focus was give to the functionalities. The UI can be made to look more appealing and sleek with powerful imaging tools like Adobe PhotoShop.

7.2.3 Dynamic Navigation

The application as of now provides directions from one place to another. It can be made to trace the current latitude and longitude while moving and directions can be updated.

7.2.4 Alternative Routes

The application provides only a single choice of directions. It can be enhanced to provide alternate routes or to provide walking directions.

7.2.5 Include More Choices

The application can be extended to provide more features such as displaying coupons or promotions if available for any of the places, which will make GoBuddy more worthy and attractive.

APPENDIX A

SAMPLE SOURCE CODE

SAMPLE SOURCE CODE
GBfavsearch.java

```
/** GoBuddy Android Mobile Application
    This component was designed for GoBudddy Android mobile application
    @author Kalaivani Nellaiappan
    Date: 5th Nov 2010 */

package com.sample.gobuddy;

import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.widget.Toast;
import android.app.Activity;
import org.apache.http.*;
import org.apache.http.client.*;
import org.apache.http.HttpEntity;
import org.apache.http.HttpRequest;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.HttpClient;
import org.apache.http.impl.client.DefaultHttpClient;
import java.io.*;
import android.widget.TextView;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;
import org.xml.sax.InputSource;
import org.apache.http.util.EntityUtils;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;
import android.content.Intent;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
```

```
import com.google.android.maps.Overlay;
import com.google.android.maps.OverlayItem;
import com.google.android.maps.MapView.LayoutParams;
import com.google.android.maps.GeoPoint;
import java.util.List;
import android.graphics.drawable.Drawable;
import android.widget.LinearLayout;
import android.view.View;
import android.widget.Button;
import com.google.android.maps.MapActivity;
import android.webkit.WebView;
import android.content.Context;

public class GBfavsearch extends MapActivity {


/** Called when the activity is first created. */

TextView ttitle,taddress,tcity,tv,disttv;
Button distbackbutton;
TextView titleplace;
Double latival,longival;
Double targetlati,targetlongi;
String direction,direction3,direction4;
String phonenum="";
String pageurl="";
String pageclickurl="";
int totalresults=0;
Buttonnext,previous,call,url,address,back,direct,b,addback;
Button sat,traffic,street;
int n=0;
GeoPoint geopoint;
MapView gbmapView;
MapController mc;
WebView wview;
List<Overlay> mapOverlays;
Drawable drawable,drawable1,drawable2;
marker mark,mark1,mark2;
int ch;
int r;
int distval;
int durval;
```

```
myapp mapp;
Location loca,locb;
Double lata,longa,latb,longb;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);
setContentView(R.layout.maps);
mapp=((myapp)getApplicationContext());
ch=mapp.getchoices();
mapp.setdirectionschoice(1);
Toast t15=Toast.makeText(getApplicationContext(),"Choice"+ch,
Toast.LENGTH_LONG);
t15.show();
gbmapView = (MapView)findViewById(R.id.mapview);
LinearLayout zoomLayout = (LinearLayout)findViewById(R.id.zoom);
View zoomView = mapView.getZoomControls();
zoomLayout.addView(zoomView,
new LinearLayout.LayoutParams(
LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT));
gbmapView.displayZoomControls(true);
mapOverlays = mapView.getOverlays();
drawable = this.getResources().getDrawable(R.drawable.icon1);
mark = new marker(drawable,getApplicationContext());
mc=mapView.getController();
Double lati=new Double("34.062898");
Double longi=new Double("-117.254248");
GeoPoint gpint=new GeoPoint((int)(lati*1E6),(int)(longi*1E6));
mc.animateTo(gpint);
mc.setCenter(gpint);
mc.setZoom(17);

OverlayItem overlayitem = new OverlayItem(gpint, "", "");
mark.addOverlay(overlayitem);
mapOverlays.add(mark);
String getgp=mark.getCenter().toString();
next=(Button)findViewById(R.id.nextbutton);
previous=(Button)findViewById(R.id.prevbutton);
sat=(Button)findViewById(R.id.satbutton);
traffic=(Button)findViewById(R.id.trafficbutton);
```

```
street=(Button)findViewById(R.id.streetbutton);
back=(Button)findViewById(R.id.backbutton);
titleplace=(TextView)findViewById(R.id.texttitle);
find(ch);
dispText(n);
dispMap(mapp.getgeopoint(n));
back.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
mapp.setchoices(ch);
startActivity(new Intent(favsearch.this,gobuddy.class));
}       });
next.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
n=n+1;
if(n<totalresults){
gbmapView.invalidate();
dispText(n);
dispMap(mapp.getgeopoint(n));
}
else{
Toast t20=Toast.makeText(getApplicationContext(),"No more results.Search
        New criteria",Toast.LENGTH_LONG);
t20.show();
}
}});
previous.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
n=n-1;
if(n>=0){
gbmapView.invalidate();
dispText(n);
dispMap(mapp.getgeopoint(n));
}
else{
        Toast t21=Toast.makeText(getApplicationContext(),"First
        result",Toast.LENGTH_LONG);
t21.show();
}
}
});

sat.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
```

```
mc.setCenter(mapView.getMapCenter());
gbmapView.setSatellite(true);
gbmapView.setStreetView(false);
gbmapView.setTraffic(false);
mc.setZoom(12);
mc.stopPanning();
}});
traffic.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
mc.setCenter(mapView.getMapCenter());
gbmapView.setTraffic(true);
gbmapView.setSatellite(false);
gbmapView.setStreetView(false);
mc.setZoom(12);
mc.stopPanning();
}});
street.setOnClickListener(new View.OnClickListener() {
public void onClick(View v) {
mc.setCenter(mapView.getMapCenter());
gbmapView.setStreetView(true);
gbmapView.setSatellite(false);
gbmapView.setTraffic(false);
mc.setZoom(12);
mc.stopPanning();
}});
}
protected boolean isRouteDisplayed(){
return true;
}
public void find(int c){
String Yahooapi="";
        LocationManager GBlocationManager = (LocationManager)
                this.getSystemService(Context.LOCATION_SERVICE);
LocationListener GBlocationListener = new LocationListener() {
public void onLocationChanged(Location gblocation) {
}
public void onStatusChanged(String gbprovider, int gbstatus, Bundle gbextras) {}
public void onProviderEnabled(String gbprovider) {}
public void onProviderDisabled(String gbprovider) {} };
GBlocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
GBlocationListener);
String GBlocationProvider = LocationManager.GPS_PROVIDER;
```

```
Location GBl=new Location(GBlocationProvider);
la = GBl.getLatitude();
lg=GBl.getLongitude();
switch(c)
{
case 0:{

Yahooapi="http://local.yahooapis.com/LocalSearchService/V3/localSearch?appid=m4yBf
cjV34Gw0G6GjgdcpooVS6lgu6BGHrGiFmJm2xjyDYYZ9xd5AazNdATvgw--
&query=coffee&latitude=34.056291&longitude=-117.182339";
break;
}
case 1:{

Yahooapi="http://local.yahooapis.com/LocalSearchService/V3/localSearch?appid=m4yBf
cjV34Gw0G6GjgdcpooVS6lgu6BGHrGiFmJm2xjyDYYZ9xd5AazNdATvgw--
&query=Restaurant&latitude=34.056291&longitude=-117.182339";
break;
}
case 2:{

Yahooapi="http://local.yahooapis.com/LocalSearchService/V3/localSearch?appid=m4yBf
cjV34Gw0G6GjgdcpooVS6lgu6BGHrGiFmJm2xjyDYYZ9xd5AazNdATvgw--
&query=fast+foods&latitude=34.056291&longitude=-117.182339";
break;
}
case 3:{

Yahooapi="http://local.yahooapis.com/LocalSearchService/V3/localSearch?appid=m4yBf
cjV34Gw0G6GjgdcpooVS6lgu6BGHrGiFmJm2xjyDYYZ9xd5AazNdATvgw--
&query=gas+station&latitude=34.056291&longitude=-117.182339";
break;
}
case 4:{

Yahooapi="http://local.yahooapis.com/LocalSearchService/V3/localSearch?appid=m4yBf
cjV34Gw0G6GjgdcpooVS6lgu6BGHrGiFmJm2xjyDYYZ9xd5AazNdATvgw--
&query=Hotel&latitude=34.056291&longitude=-117.182339";
break;

}
case 5:{
```

```
Yahooapi="http://local.yahooapis.com/LocalSearchService/V3/localSearch?appid=m4yBf
cjV34Gw0G6GjgdcpooVS6lgu6BGHrGiFmJm2xjyDYYZ9xd5AazNdATvgw--
&query=Movie+Theater&latitude=34.056291&longitude=-117.182339";
break;

}
case 6:{

Yahooapi="http://local.yahooapis.com/LocalSearchService/V3/localSearch?appid=m4yBf
cjV34Gw0G6GjgdcpooVS6lgu6BGHrGiFmJm2xjyDYYZ9xd5AazNdATvgw--
&query=Attraction&latitude=34.062898&longitude=-117.254248";
break;

}

default:Toast.makeText(favsearch.this, "Have a Nice Day",
Toast.LENGTH_SHORT).show();
}
String GBurl=String.format(Yahooapi);
HttpGet Gbget = new HttpGet(GBurl);

HttpClient GBclient = new DefaultHttpClient();
HttpContext GBcont=new BasicHttpContext();
HttpResponse GBresponse;
String GBhttpresponse="";
try {
HttpEntity GBentity = GBresponse.getEntity();

if (entity != null) {
GBhttpresponse = EntityUtils.toString(GBentity);

DocumentBuilderFactory GBdbf = DocumentBuilderFactory.newInstance();

DocumentBuilder GBdb = GBdbf.newDocumentBuilder();

Document GBdom = GBdb.parse(new InputSource(new StringReader(GBhttpresponse)));

NodeList noderset=dom.getElementsByTagName("Result");
Node resultset=noderset.item(0);
totalresults=noderset.getLength();
for(int j=0;j<noderset.getLength();j++){
Node n=noderset.item(j);
NodeList noderesult=n.getChildNodes();
```

```
Element eltitle=(Element)noderesult.item(0);
String stitle=eltitle.getFirstChild().getNodeValue();
mapp.setplacetitle(j,stitle);

Element eladdress=(Element)noderesult.item(1);
String saddress=eladdress.getFirstChild().getNodeValue();
mapp.setplaceaddress(j,saddress);


Element elcity=(Element)noderesult.item(2);
String scity=elcity.getFirstChild().getNodeValue();
mapp.setplacecity(j,scity);

Element elstate=(Element)noderesult.item(3);
String sstate=elstate.getFirstChild().getNodeValue();
mapp.setplacestate(j,sstate);

Element elphone=(Element)noderesult.item(4);
String sphone=elphone.getFirstChild().getNodeValue();
mapp.setplacephone(j,sphone);

Element ele_lat=(Element)noderesult.item(5);
String lat=ele_lat.getFirstChild().getNodeValue();
la=new Double(lat);
mapp.setlatval(j,la);

Element ele_lng=(Element)noderesult.item(6);
String lng=ele_lng.getFirstChild().getNodeValue();
lg=new Double(lng);
mapp.setlngval(j,lg);

mapp.setgeopoint(j,new GeoPoint((int)(la *1E6),(int)(lg *1E6)));
Element elurl=(Element)noderesult.item(12);
String surl=elurl.getFirstChild().getNodeValue();
mapp.setplaceurl(j,surl);

Element elclickurl=(Element)noderesult.item(10);
String sclickurl=elclickurl.getFirstChild().getNodeValue();
mapp.setplaceclickurl(j,sclickurl);

}
```

```
NodeList noderating=dom.getElementsByTagName("Rating");
for(int k=0;k<noderating.getLength();k++){
Node nrate=noderating.item(k);
NodeList noderatings=nrate.getChildNodes();

Element elrating=(Element)noderatings.item(0);
String srating=elrating.getFirstChild().getNodeValue();
mapp.setplacerating(k,srating);
}

}
catch(ParserConfigurationException pce) {
Toast t1=Toast.makeText(getApplicationContext(),"PCE",Toast.LENGTH_LONG);
t1.show();
pce.printStackTrace();
}
catch(SAXException se) {
Toast t2=Toast.makeText(getApplicationContext(),"SAX",Toast.LENGTH_LONG);
t2.show();
se.printStackTrace();
}
catch(IOException ioe) {
Toast t3=Toast.makeText(getApplicationContext(),"IO",Toast.LENGTH_LONG);
t3.show();
ioe.printStackTrace();
}
}

public void updatevalue(int x){
r=x;
mapp.setindex(x);
mapp.setaddresses(x,mapp.getplaceaddress(x)+","+mapp.getplacecity(x));
phonenum=mapp.getplacephone(x);
pageurl=mapp.getplaceurl(x);
pageclickurl=mapp.getplaceclickurl(x);
lativa1=mapp.getlatval(x);
longival=mapp.getlngval(x);
geopoint=new GeoPoint((int) (mapp.getlatval(x) * 1E6), (int) (mapp.getlngval(x) *
1E6));
titleplace.setText(mapp.getplacetitle(r));
}

public void displayMap(GeoPoint gpt){
```

```
for(int i=1;i<mapOverlays.size();i++){
mapOverlays.remove(i);
}

drawable1 = this.getResources().getDrawable(R.drawable.mapmarkers);
mark1 = new marker(drawable1,getApplicationContext());
mc.animateTo(gpt);
mc.setCenter(gpt);
mc.setZoom(12);

drawable2 = this.getResources().getDrawable(R.drawable.searloc);
OverlayItem overlayitem = new OverlayItem(gpt, "Hi", mapp.getplacetitle(r));
mark1.addOverlay(overlayitem);
mapOverlays.add(mark1);
mc.stopPanning();
String gtgp=mark1.getCenter().toString();
}
}
```

APPENDIX B

SAMPLE TEST CASES AND RESULTS

SAMPLE TEST CASES AND RESULTS

TEST CASE : To check if the value returned by a touch event or key press o the grid view is the same as expected.

Name of the Test Case Project : GoBuddyTesting

Name of the test case file – GoBuddyTest

A new Android Junit testing project was created. This test case is used to check the function of the UI Component Grid View. So the class file extends the ActivityInstrumentationTestCase. The setup() method is used to initialize the variables and the testPreConditions() is used to check if the initialization was done properly. The UI test case was written to check if the item selected on the grid view of the main screen displayed the expected value of choice, since the application's core functionality depends on this choice.

Source Code for TestCase – GoBuddyTest.java

```
package com.sample.gobuddy.test;

import android.test.ActivityInstrumentationTestCase2;
import com.sample.gobuddy.gobuddy;
import android.app.Activity;
import android.view.KeyEvent;
import android.widget.GridView;
import com.sample.gobuddy.ImageAdapter;
public class GoBuddyTest extends ActivityInstrumentationTestCase2<gobuddy> {
private Activity gbtestactivity;
private static final int icons=8;
private static final int initial=0;
private static final int testposition=4;
private GridView gridview;

private ImageAdapter imgadapter;
public GoBuddyTest(){
```

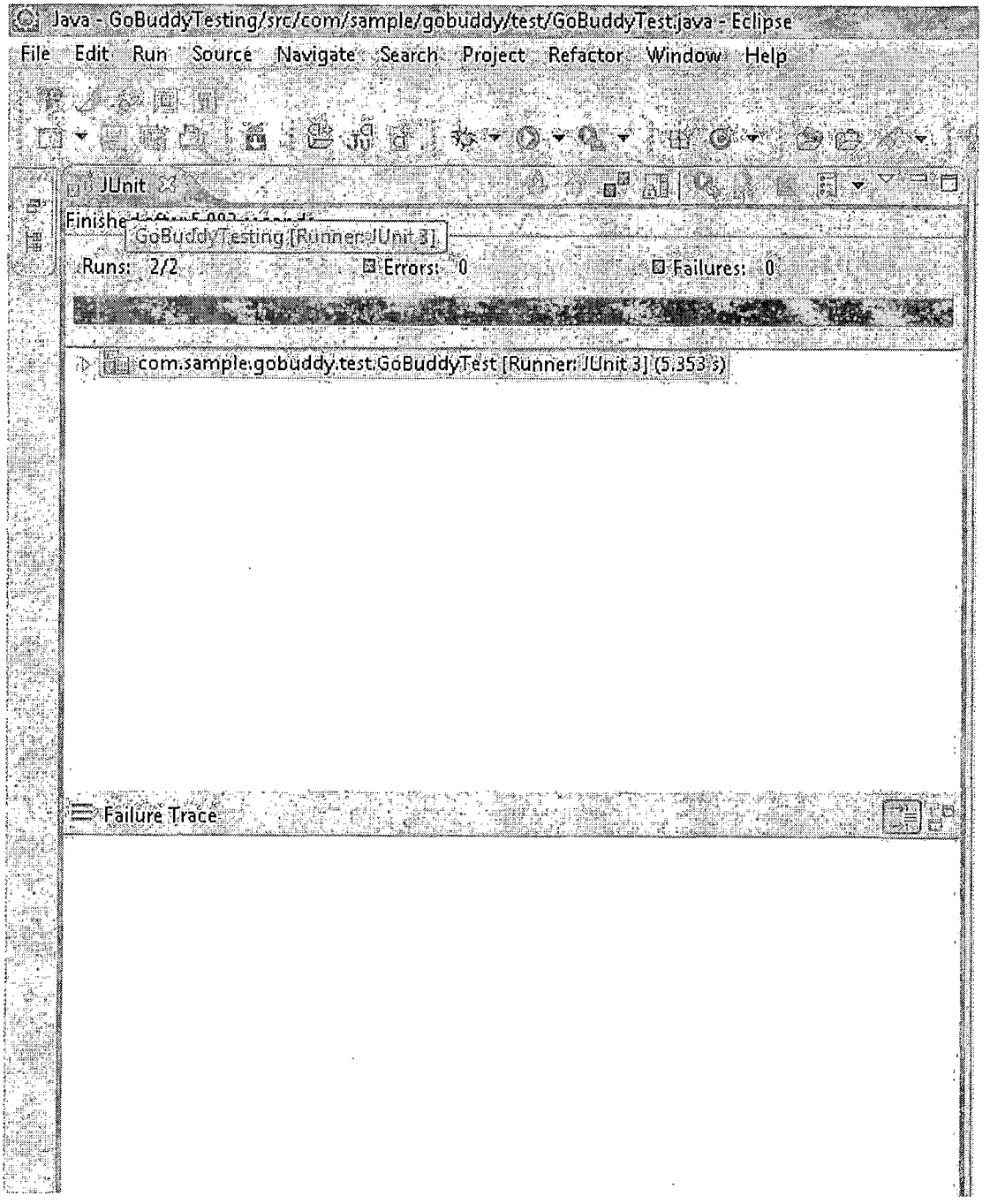
```
super("com.sample.gobuddy",gobuddy.class);
}

@Override
protected void setUp() throws Exception {
super.setUp();
gbtestactivity = this.getActivity();
gridview=(GridView)gbtestactivity.findViewById(com.sample.gobuddy.R.id.g
        ridview);
imgadapter=(ImageAdapter)gridview.getAdapter();
}
public void testPreConditions() {
assertTrue(gridview.getOnItemClickListener()!= null);
assertTrue(imgadapter != null);
assertEquals(gridview.getCount(),icons);
}
public void testGridSelection(){
gbtestactivity.runOnUiThread(
new Runnable() {
public void run() {
gridview.requestFocus();
gridview.setSelection(initial);
}
});
for(int i=1;i<=2;i++){
this.sendKeys(KeyEvent.KEYCODE_DPAD_DOWN);
}
int currpos=gridview.getSelectedItemPosition();
assertEquals(currpos,testposition);
}
}
```

The application was run as Android Junit test and the following results were obtained.

Android Console displaying the GoBuddyTesting ran successfully

Test Case results showing a successful test run

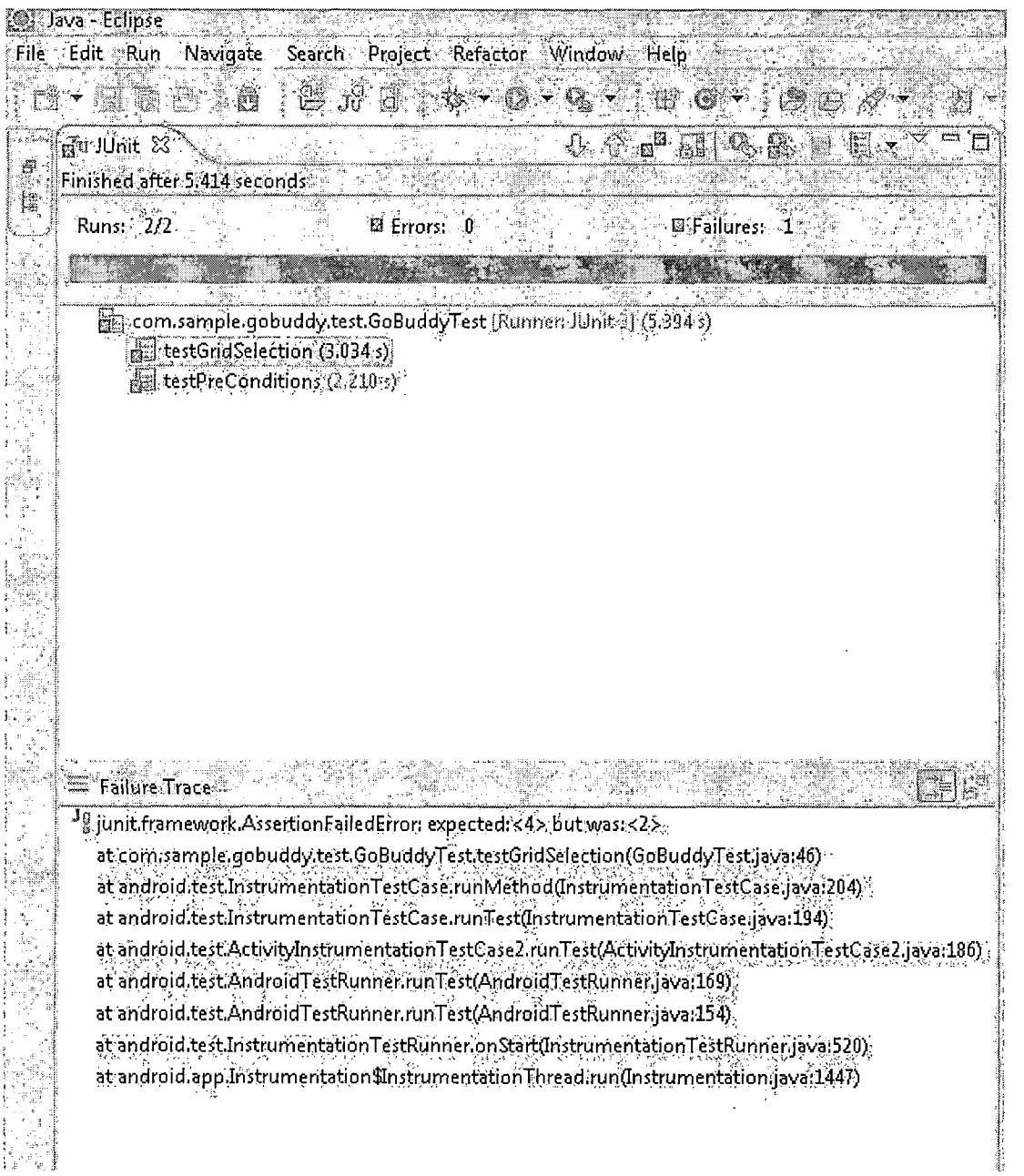
Junit Window



Test Case results where the expected value didn't exactly match the choice returned by the item selected on the grid view

Test Case showing a failure test run

Junit Window



REFERENCES

[1] Wikimedia Foundation Inc, "Smart Phone", wikipedia.org, 16 July. 2010. [Online]. Available: http://en.wikipedia.org/wiki/Smartphone [Accessed: 20 Sep. 2010].

[2] Google Inc, "Places App", google.com, [Online]. Available: http://www.google.com/support/mobile/bin/answer.py?hl=en&answer=186393 -Places app [Accessed: 15 Sep. 2010].

[3] Poynt Corporation, "Poynt", poynt.com, [Online]. http://www.poynt.com/whyBenefits.html [Accessed: 15 Sep. 2010].

[4] Wikimedia Foundation Inc, "Android", wikipedia.org, 10 Nov. 2009. [Online]. Available: http://en.wikipedia.org/wiki/Android_(operating_system) [Accessed: 11 Dec. 2010].

[5] Wikimedia Foundation Inc, "Software Development Kit", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Software_development_kit [Accessed: 11 Dec. 2010].

[6] Google Inc, "Android SDK", android.com, 13 Oct. 2010 [Online]. Available: http://developer.android.com/guide/basics/what-is-android.html [Accessed: 10 Dec. 2010].

[7] Wikimedia Foundation Inc, "Dalvik", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Dalvik_(software) [Accessed: 10 Dec. 2010].

[8] Wikimedia Foundation Inc, "Object Oriented Programming", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Object-oriented_programming [Accessed: 11 Dec. 2010].

[9] Refsnes Data, "XML Parser", w3schools.com, [Online]. Available: http://www.w3schools.com/xml/xml_parser.asp [Accessed: 19 Dec. 2010].

[10] Google Inc, "Android SDK", android.com, 13 Oct. 2010 [Online]. Available: http://developer.android.com/sdk/installing.html [Accessed: 28 Dec. 2010].

[11] Wikimedia Foundation Inc, "Eclipse software", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Eclipse_(software) [Accessed: 28 Dec. 2010].

[12] Wikimedia Foundation Inc, "Hyper Text Transfer Protocol", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol [Accessed: 2 Dec. 2010].

[13] QuinStreet Inc, "HTTP", internet.com, [Online]. Available: http://www.webopedia.com/TERM/H/HTTP.html [Accessed: 15 Dec. 2010].

[14] Wikimedia Foundation Inc, "Dia", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Dia_(software) [Accessed: 11 Dec. 2010].

[15] Wikimedia Foundation Inc, "Unified Modeling Language", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Unified_Modeling_Language [Accessed: 22 Dec. 2010].

[16] The GIMP Team, "GIMP", gimp.org, [Online]. Available: http://www.gimp.org/about/introduction.html [Accessed: 15 Dec. 2010].

[17] Google Inc, "Google Directions API", google.com, [Online]. Available:http://code.google.com/apis/maps/documentation/directions/ [Accessed: 29 Nov. 2010].

[18] Wikimedia Foundation Inc, "Document Object Model", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/XML#Document_Object_Model_.28DOM.29 [Accessed: 28 Dec. 2010].

[19] Fowler, M.,"UML DISTILLED", (Third Edition)-Covers through Version 2.0 OMG UML Standard.Addison-Wesly, 2004.

[20] Google Inc, "Android SDK", android.com, 13 Oct. 2010 [Online]. Available: http://developer.android.com/guide/topics/fundamentals.html [Accessed: 29 Nov. 2010].

[21] Wikimedia Foundation Inc, "Android operating system", wikipedia.org, 10 Nov. 2009. [Online]. Available: http://en.wikipedia.org/wiki/Android_(operating_system) [Accessed: 25 Dec. 2010].

[22] Wikimedia Foundation Inc, "Java", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Java_(programming_language) [Accessed: 10 Dec. 2010].

[23] Wikimedia Foundation Inc, "XML", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/XML [Accessed: 28 Dec. 2010].

[24] Wikimedia Foundation Inc, "Application Programming Interface", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Application_programming_interface [Accessed: 1 Dec. 2010].

[25] Wikimedia Foundation Inc, "Google Maps", wikipedia.org, [Online]. Available: http://en.wikipedia.org/wiki/Google_Maps [Accessed: 28 Dec. 2010].